Methodology to Develop Hybrid Simulation/Emulation Model

KHALID BIN HASNAN

A thesis submitted in partial fulfillment of the requirements of
Sheffield Hallam University
for the award of
Doctor of Philosophy

December 2005

DECLARATION

This is to certify that I am responsible for the work submitted in the thesis, that the original work is my own except as specified in acknowledgements, and that neither the thesis nor the original work contained therein has been submitted to this or any institution for a higher degree.

Signature:

Name : **KHALID BIN HASNAN**

Date : **15 December 2005**

ACKNOWLEDGEMENTS

In the name of Allah, the Most Gracious, the Most Merciful.

I would like to communicate my sincere gratitude to my research supervisors, Professor Dr. Terrence Perera and Dr David R. Clegg, for their guidance and encouragement. In particular I would like to express my appreciation to Professor Terrence Perera for his interest and enthusiasm, to read, modify and comment on the manuscript.

I wish to thank all colleagues of the Manufacturing Systems Engineering Research Group particularly Anna Lassila, Tom Pohl, Saman Yapa, Piyasena Samarakoon, Ruwan Wickramarachchi, Michael Liew and Jakub Banaszak for their support to this research.

Thanks to the government of Malaysia and Kolej Universiti Teknologi Tun Hussein Onn (KUiTTHO) for sponsoring my PhD study. Special thanks to members of the Faculty of Mechanical and Manufacturing Engineering of KUiTTHO particularly to its Dean, Professor Mohd Zainal, for their kind support and cooperation towards the completion of this thesis.

Finally, and most importantly, I wish to express my love and appreciation to my wife, Suhaidah Tahir; my children Ruqayyah, Maryam, Luqman, Anisah, Nabielah, Busyra, Fatimah, Hakimi, Zarif, Izzah and Iqbal; my son-in-law Farhan and grandson Amirul Hakim; for their patience, support, encouragement and du'a throughout the work which has taken many hours that should have been dedicated exclusively to them. Also I wish to express my love to my late parents, brothers and sister as well as parent in-law for their consistent support and du'a.

ABSTRACT

Trends towards reduced life-time of products and globalised competition has increased pressure on manufacturing industries to be more responsive to changing needs of product markets. Consequently, the use of simulation to describe short term future performance of manufacturing system has become more significant than ever. An application of simulation that has attracted attention is for testing of control logic before commissioning on site by using a detailed simulation model called emulation model. However, though the success of using emulation particularly in improving cost-effectiveness of automated material handling system delivery has been acknowledged by industries and simulation model developers, the uptake for this technology is still low. The major inhibitors are the high costs of its model building as well as simulation and emulation models are perceived to be non convertible.

The main objective of this research is to establish a methodology to develop simulation model that can be converted into emulation model with ease, thus making emulation technology more affordable. The product of this research called the methodology to build Hybrid Simulation Emulation Model (HSEM) is a new approach of building emulation model comprising of three phases namely (1) development of base simulation model, (2) development of detail emulation model, and (3) integration of controller with the emulation model. Important requirements for HSEM are flexibility of adding details to the simulation model and inter process communication between model and real control system. To facilitate implementation of the methodology, it is essential that the simulation software package provide functionalities for modular model development, access and adding of codes, integration with other application and real time (RT) modelling.

The methodology developed offers a more affordable emulation modelling and an opening for further research into the comprehensive support for the implementation of real time control system testing using emulation.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Manufacturing System Development

Manufacturing is changing rapidly around the world. The processes, equipment, and systems used to design and produce everything from automobiles to computer chips are undergoing dramatic changes in response to new customer needs, competitive challenges, and emerging technologies. Advances in information systems, business practices, engineering techniques, and manufacturing science now enable companies to produce new and better products more quickly and at a much lower cost than ever before.

As a result, fundamental changes are occurring in the manufacturing environment. This can be seen in the current trend towards highly automated systems that are intended to adapt quickly to change while providing extensibility through a modular, distributed design. It has also placed the whole manufacturing system development process greater importance than ever before.

1.2 Issues on Manufacturing System Development and Delivery

In order to realise the flexibility and/or productivity that these advanced system promise, system modelling and control are viewed as vital to enable the components of these automated manufacturing systems to work together in an integrated way.

The complexity of such system imposes the need to trace products throughout the system, and include more rules and logic. Thus not only the manufacturing system design has to be efficient, the validation has to be fast and cost effective.

To understand the validation process let us look at the manufacturing system development project which generally goes through several key stages as shown in Figure 1.1.

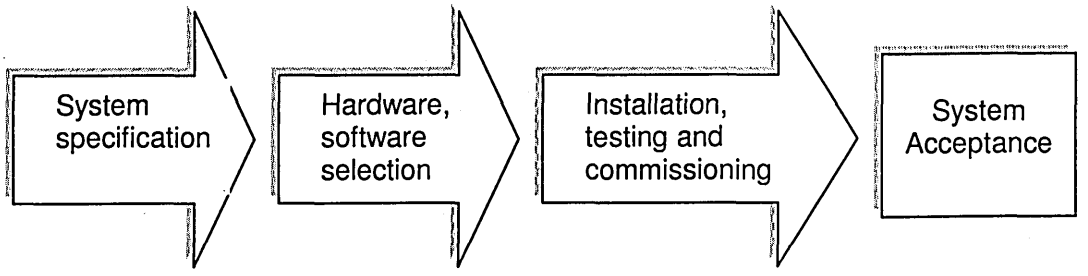


Figure 1.1 Stages for Manufacturing System Delivery

The first stage is definition and analysis, in which simulation has long been used to help determine the system specification. A well-written simulation model can be a valuable tool in the design, analysis, and operation of manufacturing and other complex systems.

The second stage is the selection of hardware and software suppliers, and the construction of the hardware. Key parts of the solution may be tested off-site, in order to verify the technology.

The third stage is on site installation, testing and commissioning. Once testing is finished, the owner takes possession of the installation and the system is ramped up. During this phase, and at any time during the working life of the installation, control system modifications may be necessary as the load on the system changes.

Commissioning covers the period of hardware installation and software testing, as well as acceptance and handoff of the project. In many cases, commissioning disrupts

existing production, and in all cases the sooner the system is successfully commissioned, the sooner the system can be used to generate revenue.

Testing the control system is an important part of the commissioning phase, and yet it is often done under extremely difficult conditions due to the following reasons:

- Some of the hardware may already be in use for production.
- Complete testing is impossible before the installation is complete.
- "Full system" control tests are often impossible.
- Control tests often conflict with hardware calibration and testing.
- Realistic current and future loading levels are unavailable for testing.
- Control software may be incomplete.
- Malfunctions may be hard to replicate.
- Modifications may be rushed and not fully tested before implementation.

The commissioning period of a project is often extended, resulting in penalties and the need to bring unbudgeted resources to the site to resolve the situation. This costs both hardware vendors and client's time and money.

Users of industrial simulation products have long requested a way to eliminate the need to reproduce the logic built into the simulation model when the experimentation and analysis phase is complete and the control system has to be developed.

1.3 Emulation: Simulation Model for Control System Testing

A growing application of simulation and communication is emulation, where a simulation model is used to replace a real Automated System in order to test and debug an industrial control system. Since an emulation model is designed to provide the same responses as the physical system, it can reliably replace the physical system for many control system tests.

Emulation and control system testing can be carried out as soon as the control system software has been developed, before commissioning. This leaves the critical commissioning window available for unavoidable hardware tests and minimises the set of control system tests that must be done during the final testing stage.

This allows control system designers to test the control logic using the simulation prior to going into the field, which can result in considerable savings in terms of time spent on site performing routine logic testing and debugging as illustrated in Figure 1.2.

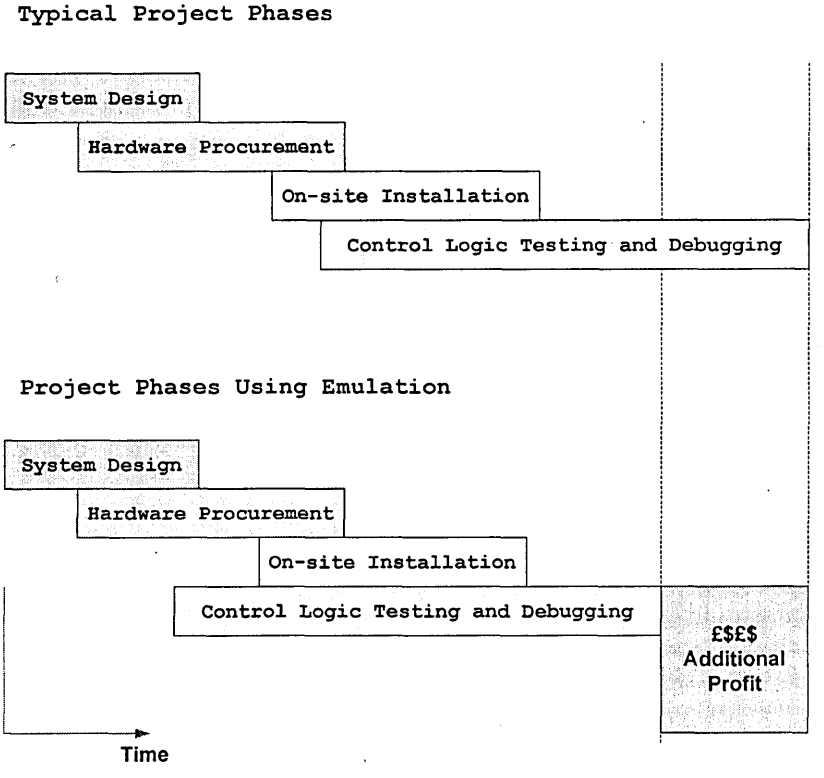


Figure 1.2 Time and cost benefits of emulation

Emulation provides a reliable and safe way of verifying control code functionality offline, training operators in a safe environment, and of testing modifications to a control system before they get put into effect. Furthermore, the emulation model serves as a test bed for any further control system modifications throughout the life of

the system, so production is not disrupted. Being a software solution, it is easier to implement and modify than hardwired panels. It can also be stored for reuse on a future project and provides the possibility of early testing.

Another advantage of emulation model is that it can be maintained for the lifetime of the automated system it represents. During the productive phase of the system, any proposed changes can be tested on the model before they are implemented on the shop floor, just as the original design was tested. This can save money and provide greater returns from the system, because it can be adapted to keep pace with business changes. The updated model can also be used for training employees about changes to the system, as well.

1.4 Developing Hybrid Simulation-emulation model

Even though the benefits of using emulation for the analysis of manufacturing control systems are well acknowledged, the speed and cost of the model building remains a concern.

At present, development of an emulation model has to be done independently from simulation model. In other words, a project will require a simulation model for initial analysis and development as well as a separate emulation model for testing a control system. The main reasons for requiring the two-stage development are that an emulation model is often more detailed than a simulation and also emulation model must include communication logic.

Even so, the development of separate software logic for all levels of detail would cause duplication of effort which renders it not cost effective and also creates difficulty in maintaining consistency.

A proposed solution is to develop a methodology to build hybrid simulation-emulation model or composable simulation model, one that is used for both purposes

and should have a facility to switch off/on certain elements from the model as necessary. The present work investigates this possibility and proposes a system approach towards its development.

1.5 Summary

This chapter has introduced the role of discrete event simulation in manufacturing system development. The importance of efficient validation of manufacturing system design is highlighted. Emulation model, a new form of simulation model developed for early validation of control system is introduced. The chapter ends with underlining the aim of the present research which is to develop a rapid development approach for emulation model building.

The outline for proceeding chapters is as below.

Chapter 2 reviews in more detail the previous works related to emulation and outlays the aims and objectives of the present research.

Chapter 3 describes the justification of the research and the research methodology employed in this research.

Chapter 4 gives the requirement specification for simulation-emulation model building.

Chapter 5 describes the proposed simulation to emulation model conversion methodology.

Chapter 6 describes the validation of the methodology.

Chapter 7 concludes with the findings of the research and propose recommendation for future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The previous chapter has introduced the concept of emulation modelling and its benefits. Special emphasis has been drawn towards the need to build the emulation model more cost effective. This chapter presents a background of emulation technology in more detail based on case studies of some emulation-related manufacturing projects.

It begins with defining the scope of investigation for the proposed research. This is followed by a review on the use of simulation in a variety of applications in manufacturing industry. The next section looks at Discrete Event Simulation (DES), the most widely used type of simulation in manufacturing, particularly in the design and testing of the manufacturing system. The discussion is then directed towards a new form of DES called emulation and its role in control system testing, which become the focus of this thesis. This section is followed by comparative study between simulation and emulation model. It also reports the investigation on the current applications of emulation models in manufacturing industry and the approaches that the models were developed in order to establish the specific requirements for building emulation model from simulation model. The chapter concludes by summarising the findings of the case studies.

2.2 Scope of investigation

Initial literature search has indicated that the term 'emulation' has also been used in a variety of application area and context. For example it is also used in the computer gaming community, software (programming) emulators as well as electronic hardware emulators. However, to avoid ambiguity and to maintain consistency in the context of the proposed research the term 'emulation' was referred to as a new form of discrete event simulation being used to model the plant in manufacturing system design and validation.

From the initial search it was also realised that literature on 'emulation' particularly in regard to its model building was very limited. Consequently, the search was broadened to include the use of the phrases 'simulation for control system testing' 'real time simulation', as well as ' flexibility of adding details' which to some extent considered related to the context of the proposed research.

'Simulation for control system testing' relates to the purpose of emulation in this research context (Habchi and Berchet 2003; Shnits et al. 2004) . 'Real time simulation' provides the understanding of the technology concerning real time communication from emulation(Dougall 1998; Jeong and Kim 1998; Julia and Valette 2000; Stewart et al. 2003). And, ' flexibility of adding details' relates to the multi-resolution modelling as well as usefulness and credibility of the emulation model (Ball 1998; Persson 2002).

Thus, the scope of investigation, within the defined context covered the technologies and development methodologies related to the above mentioned phrases together with the literature search for the relatively new technology of emulation.

2.3 Simulation Technologies in Manufacturing

In general, manufacturing system design involves making long-term decisions such as facility layout and system capacity/configuration. As such, models are typically created and used for a single design exercise, and model run time is not a significant factor during the simulation process.

Manufacturing system operation, on the other hand, involves making decisions on a much shorter time schedule. The activities include operations planning and scheduling, real-time control, operating policies, and performance analysis As such, the model is generally used (and reused) much more frequently, and simulation run time is a more significant factor in the software/package selection and model design process (Smith 2003).

Modelling of such manufacturing systems can be achieved using a number of tools and techniques. While modelling and analysis are important to help ensure good system performance, the integration and complexity of manufacturing systems often makes purely analytic tools difficult to use. Hence, simulation remains one of the most widely used tools to fill this need.

One of the most quoted description of simulation, (Banks 1998) defined simulation as "the imitation of the operation of a real-world process or system over time. Simulation involves the generation of an artificial history of the system, and the observation of that artificial history to draw inferences concerning the operating characteristics of the real system that is represented".

Or put another way, simulation is the technique of a building a model of a real or proposed system so that the behaviour of the system under specific conditions may be studied. One of the key powers of simulation is the ability to model the behaviour of a system as time progresses.

Various simulation technologies and software for manufacturing are available and selection is normally based on the application area and purpose of study apart from the monetary and time constraints.

Within the scope of manufacturing systems the general application areas of simulation include business process modelling, manufacturing process modelling, supply chain modelling, and process and system visualisation

Focusing more specifically on *production systems*, there are a large number of application areas and simulation technologies. Some simulation technologies are specific to certain application. For example, human tasks simulation is used for ergonomic studies of work areas and manual tasks. Robotic simulation deals with motion and collision control of industrial robots as well as of-line programming of equipment including industrial robots. Assembly simulation is used as an aid for process planning for assembly operation, accessibility and to investigate assembly feasibility (Banks et al. 2000).

A more generic discrete event simulation is widely used in facility design, material handling system design, manufacturing cell design, and flexible manufacturing system (FMS) design. The activities include buffer sizing, lot sizing, material flow including bottleneck detection, plant layout effects, scheduling evaluations, costs and work in process levels (Holst 2001). A more specific form of discrete event simulation called emulation is used to develop and validate control strategies for automated system like AMHS (automated material handling systems) of which AGVS (automated guided vehicle systems) and ASRS (automated storage and retrieval systems) are examples.

2.3 Discrete Event Simulation

Discrete event simulation is one way of building up models to observe the time based (or dynamic) behaviour of a system. A discrete-event simulation is one in which the

state of a model changes at only a discrete, but possibly random, set of simulated time points. During the experimental phase the models are executed (run over time) in order to generate results. The results can then be used to provide insight into a system and a basis to make decisions on. As for emulation, due to the fact that it involves interaction with controller which operates in discrete manner, it is imperative that Discrete Event Simulation is taken as the foundation for its model building. Thus it is essential to understand basic concepts of discrete event simulation.

2.3.1 Key Elements of Discrete-Event Simulation Software

The process of building simulation models will invariably involve some form of software. The software could either be a high level programming language or a data driven software system in which the model is specified using user-defined and default data items. Hence the model is either the software itself or held within a host software system. With the development of more user-friendly simulation systems it is generally the user who will build the model, not an expert.

Inside the software or model will be a number of important concepts, namely entities and logic statements. Entities are the tangible elements found in the real world, e.g. for manufacturing these could be machines or trucks. The entities may be either temporary (e.g. parts that pass through the model) or permanent (e.g. machines that remain in the model). The concepts of temporary and permanent are useful aids to understand the overall objectives of using simulation, usually to observe the behaviour of the temporary entities passing through the permanent ones.

Logical relationships link the different entities together, e.g. that a machine entity will process a part entity. The logical relationships are the key part of the simulation model; they define the overall behaviour of the model. Each logical statement (e.g. "start machine if parts are waiting") is simple but the quantity and variety and the fact that they are widely dispersed throughout the model give rise to the complexity.

Another key part of any simulation system is the simulation executive. The executive is responsible for controlling the time advance. A central clock is used to keep track of time. The executive will control the logical relationships between the entities and advance the clock to the new time. The process is illustrated in Figure 2.1. The simulation executive is central to providing the dynamic, time based behaviour of the model. Whilst the clock and executive are key parts of a simulation system they are very easy to implement and are extremely simple in behaviour.

Two other elements that are vital to any simulation system are the random number generators and the results collation and analysis. The random number generators are used to provide stochastic behaviour typical of the real world. For example, machine scrap rates will rarely be fixed but will vary between certain ranges hence the scrap rate of a machine should be determined by a random distribution.

The results collation and display provides the user a means of utilising the simulation tool to provide meaningful analysis of the new or proposed system. Simulation tools will typically display tabulated raw results and possess some graphing capabilities.

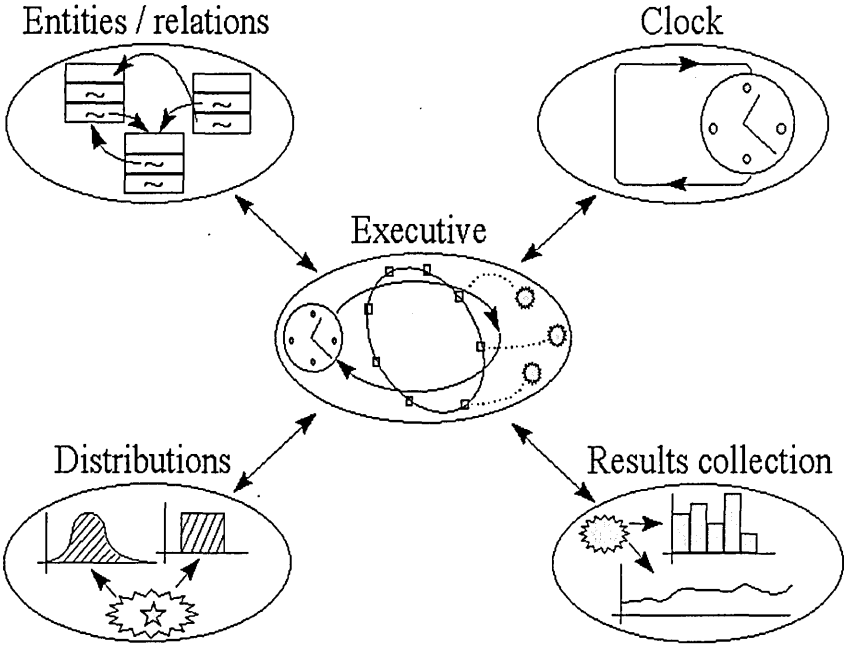


Figure 2.1 Structure of a simulation system (Ball 1996)

Today, there are hundreds of commercially available DES software packages; some based on the Simulation Programming Languages (SPL) , some on general programming languages, and yet others on proprietary SPLs; some are 2-D, others come in 3-D, and a few offer both; and they range in price from a few hundred pounds to tens of thousands of pounds.

These simulation packages can be further classified into general-purpose simulation packages and application-oriented simulation packages (Law and Mccomas 1999), meaning that they differ in their area of application, from very general (such as Extend and Simul8) to highly specialized packages for various manufacturing applications (such as AutoMod and Quest), or call centres (such as Arena's .Contact Center Edition.), just to mention a few. In fact, the level of specialization in manufacturing goes even further, as evidenced by for example automated storage and retrieval system (AS/RS) modules (such as for Quest).

With regards to emulation modelling in the context of manufacturing, a specific discussion on the software requirement is presented in subchapter 4.4

2.3.2 Discrete-Event Simulation and Control System Testing

As been highlighted in the previous section, control system testing using simulation has received considerable attention among the simulation practitioners (Banks 2000). In line with the scope of the present thesis, the discussion is focused towards manufacturing system development.

The development of a complex real system which is controlled by a control system may include one or more of the following four design stages or testing types which is illustrated in figure 2.2 (Auinger et al. 1999).

Full prototyping, shown as type A, involves tests with real equipment or system to be controlled and real control systems. This seems the most realistic testing possibility,

although it is quite expensive to build and experiment with the whole prototype system, especially because it involves the risk of failures if the possibilities of its design are not tested thoroughly beforehand.

Full simulation or offline simulation, shown as type B, on the other hand, does not involve so high costs. However, it may disregard some phenomena that are present in the real system or contain additional factors that might influence the outcomes. Among these are the issues arising from the fact that the control architecture is usually distributed across a network of computers and communication requirements among the distributed computing processes are a major concern. It is often difficult to adequately model the communications requirements using software alone. Furthermore, it is often difficult to foresee all potential deadlock situations that can arise and include these within the software simulation of the system.

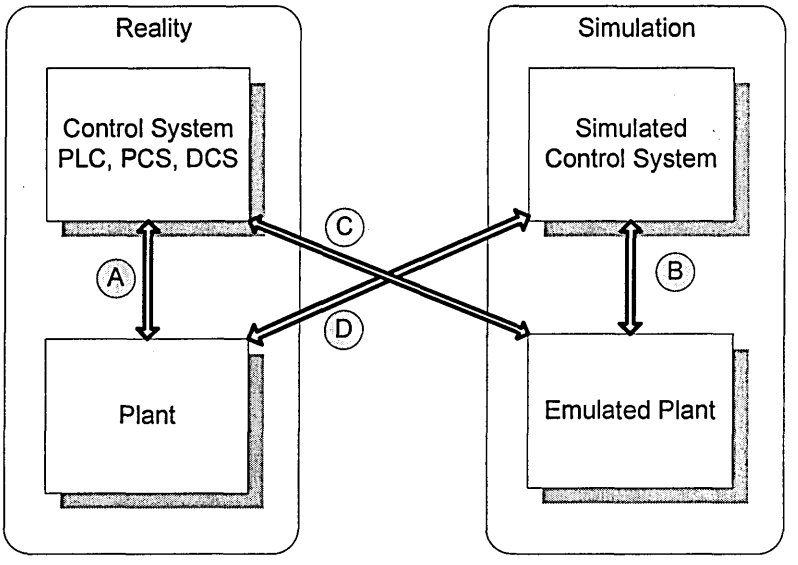


Figure 2.2 Possible Combinations between Reality and Simulation for Control System Testing.
(Adapted from (Auinger et al. 1999))

Type C which represents the context of this research emulates the equipment or system to be controlled and uses real control systems. Also called hardware-in-the-

loop (HIL)-based approach where the inputs and outputs of a controller are connected to a simulation of the part to be controlled (Rabbath et al. 2000). The detailed emulated model is also called emulation (Mueller 2001; Schiess 2001).

Reality in the loop or Real-time control shown as type D uses real equipment and simulates the control systems. This type of testing is also called test bench testing as the equipment to be tested is usually relatively small and easy to set up but the control logic could be complex.

It is important to note that both the development of the real system and the development of the software control system are very expensive. Emulation and real-time control have the advantage that they can be carried out in a cheaper way than full prototyping, and stay closer to reality.

Other advantages of emulation as pointed by (Mcgregor 2000) include :

(1) Emulation allows earlier, more complete testing.

(2) Emulation allows more time to modify and repair control code.

(3) Emulation helps improve client relations.

(4) Emulation facilitates inexpensive, non-disruptive operator training.

(5) Emulation provides a safe means of testing system modifications off-line.

However, building an emulation model is still unaffordable for small and medium scale industries, as could be seen from the examples of emulation projects discussed in section 2.5. This is mainly due to its complex nature and as noted in the literature search and also from the discussion with simulation software suppliers and users, that at present an emulation model has to be built independently from simulation model. To find ways of reducing the cost, the characteristics and requirements of emulation model need to be better understood. The next section reviews the similarities and differences between simulation and emulation model.

2.4 Comparisons between Simulation and Emulation Models

Although a simulation model and an emulation model may look the same, and may be built largely with the same building blocks, there are significant differences in usage as well as their operation. The operational differences include the execution clock, inclusion of hardware, level of detail, system coding, external communication and repeatability.

2.4.1 Different Objectives

Simulation models are used to test and develop different solutions in order to arrive at a better solution, based on an accepted set of pre-defined metrics. It often provides the impartial judge between experience and new ideas, and allows the user to demonstrate functionality and results in a cost-effective and flexible environment. Simulations results help define the physical layout of a system, its operating limits and its control system. Models are used as a basis for extensive experimentation, often using automatic procedures to determine optimal or robust solutions. (Mehrabi et al. 2000)

As the aim of a simulation model is exploratory by nature, the faster it can cover all different possibilities, the better. Simulation modelling software is therefore designed and developed with speed of execution in mind, and the models built with it are also often constructed for fast execution.

(Rohrer and McGregor 2002) argued that emulation models are used in a much more precisely defined way; in order to test the operation of the control system under different system loading conditions, and as a risk-free means of training system operators and maintenance staff. Emulation models are not used for experimentation in the same way that simulation models are; they are unsuited to this function as they often execute only in real time.

The emulation model reflects more precisely the system that will be implemented, and as such, can be used to carry out a constrained series of verification procedures to ensure the performance or reaction of the control system.

2.4.2 Hardware in the loop

A major difference between simulation and emulation is that simulation models are 'stand-alone' done all in software where as emulation models are used in conjunction with hardware, a situation also called hardware in the loop (HIL).

Good system engineering practice would begin with a pure simulation and as components become better defined (with the aid of simulation); they can be fabricated and replaced in the control loop.

For most real systems, there are characteristics that are unknown or too complex to model by pure simulation. Emulation allows hardware to be included in the model and the developers can see the real-time interactions between different hardware and software models. It is also possible to hook up real-world stimulus to peripherals and start debugging system behaviour (Wells et al. 2002).

Industrial communications networks are not deterministic, and control systems need to be designed to run reliably under varying load conditions. It therefore becomes important that emulation models be robust, like the control systems that drive industrial processes and Automated Material Handling Systems (AMHS) (Mcgregor 2002).

2.4.3 Execution clock

(Davis et al. 1996) regarded the primary difference between simulation and emulation arises with the manner in which the model is executed. A simulation model

maintains its own simulation clock. When a decision is taken within the model, the simulation clock does not advance until the necessary calculations have been performed and the decision has been evaluated. This means that simulation time stops whilst decisions are taken.

As an example consider a box on a diverging conveyor belt. In a simulation model, the box may be sent one way or the other depending on the contents of the box and its final destination. In reality, this decision may be the result of several steps, each of which takes a measurable amount of time. The box may be scanned, and a bar code read. The information may be sent via a network and used to search a database to identify the contents and the destination of the box. Then a control system may verify that a diverter is in the correct position. If it is not, then a pneumatic or electric movement takes place. The initial bar code scan will have taken place before the diverter, at a sufficient distance to ensure that the response can be calculated and the diverter moved to the appropriate position before the box arrives (Mcgregor 2002).

In essence, to be realistic an emulation model must run at exactly the same speed as the control system. Since control systems are designed to operate in real time, and so emulation experiments should be operated in real time.

Thus, the difference between an emulation and simulation in terms of execution clock may simply be summarised by saying that under a simulation, message processing procedures control the advancement of time while under an emulation, the advancement of time is controlled by a real-time clock.

The difference between logical-time simulations and real-time simulations is also apparent in their code. Logical-time simulations have the classical discrete event or continuous simulation data structures and algorithms. Real-time simulations resemble real-time systems – their execution is measured by hertz frequency, and they are typically interrupt driven (Page and Smith 1998).

While real time simulation and emulation has similar characteristics in terms of the execution clock, they are quite different in other aspects of modelling. One of the main differences is with regard to the modelling different levels of detail as explained in the next section.

2.4.4 Level of detail

To use an appropriate level of detail is one of the cornerstones of a valid model. Since the model is, by definition, an abstraction of the actual system under study, not all details are depicted in the model (Ball 1998). Choosing the appropriate level of detail seems to be a balancing act between, minimising the details on the one hand and, adding details to ensure usefulness of the model on the other hand. When reducing the level of detail, the model looses its ability to provide a useful result at some point.

An emulation model is often more detailed than a simulation model. Because the emulation model must provide the same responses to the controllers as real system hardware, the model must be designed to respond to many system events that would otherwise not require custom processing during a simulation.

Therefore the level of abstraction necessary to create the emulation should be as low as possible. For example, an emulation model might be required to send signals to a controller server when a load begins a pop-up transfer, when the transfer has completed lifting, when the load moves to the new section and when the transfer completes lowering (Mcgregor 2002).

To model every component of the system at the low level of abstraction would be inefficient as well as problematic. Models with a high level of detail or resolution describe the real system more accurately, but a simulation in a high resolution requires a long execution time. In large models with high resolution this can lead to an execution which is slower than in real time.

One way to overcome this would be through the use of parallel computing systems. Another alternative which would be much faster is to have the major part of the simulation realised on a rough level and only small parts are simulated in a high resolution hence the use of multi-resolution modelling.

2.4.5 Control system coding

A simulation model may contain control logic developed in the simulation environment that directly controls devices and loads throughout the execution of the model. An emulation model often responds to signals from the external control system, which controls system processes.

(Mcgregor 2002) argued that in order for an emulation model to operate in a way that reflects the reality of an automated system, it must be possible for the modeller to separate the physical parts of the model from the logical or operational parts. Also, in order for the modeller to experiment with the final model it may be necessary to have a part of the model operate under simulation logic, whilst other parts are under the direct control of an external control system.

(LeBaron and K.Thompson 1998) acknowledged that the main benefit of emulation is that it eliminates the need to re-implement code. Code developed and refined in traditional simulation models must be re-implemented into the actual control software if it is to be used. This creates the possibility of communication and re-implementation errors. With emulation, the actual control system is used, thus the code is developed and refined as the model is developed thus provides greater confidence in the results.

2.4.6 Inter-process communication (IPC)

Inter-process communication (IPC) is a capability supported by some operating systems that allows one process to communicate with another process. The processes

can be running on the same computer or on different computers connected through a network. IPC enables one application to control another application, and for several applications to share the same data without interfering with one another.

Off-line or full simulation often does not require IPC for the reason that it does not require communication with external application.

Emulation on the other hand must include communication logic to link the model to real time control system which consists of a continuous dialogue between sensors, control systems, and actuators. For example, in an AGV order assignment the location of each AGV is required, the driving distances or times form each location to each location should be available and the actual status of each other handlers should be available. This information is sent from different controllers such as the ASRS manager, the crane manager or the AGV manager to the external algorithm.

As such, development and use of appropriate interface between multiple real controllers and emulation model is crucial in obtaining the correct results.

2.4.7 Repeatable runs

Two or more model runs will always execute in exactly the same way and produce precisely the same results if no parameters are changed between runs. Any impression of randomness in a simulation model is due to the use of pseudorandom numbers to generate certain events such as breakdowns, cycle times and so on. Repeatability is necessary in order to recreate and understand events during the model run, as well as to debug the model as it is built. All events that influence the model execution are contained within the model and are therefore repeatable.

Due to the fact that in most emulation models the control system is separate from the model itself, repeatability is uncertain, as communication events are asynchronous and unpredictable. The model and the control system work with different clocks and

synchronize via a communications layer, itself prone to the decisions of the operating system (Mcgregor 2002).

The comparison between simulation and emulation using the above features is summarised in Table 2.1.

Table 2.1 Summary of comparison between simulation and emulation

Characteristics/features	**Simulation**	**Emulation**
Aim	To test and develop different solutions	To test control system under different conditions
Execution Clock	Virtual time	Real time
Level of detail	Low	High
Hardware in the Loop	No	Yes
Control system coding	Control rules hard coded	Control rules separated from event code developed and refined as model is developed.
External communication	Not always	Yes (interface required)
Repeatable runs	Yes (precisely same result)	No (communication events asynchronous and unpredictable)

2.5 Applications of Emulation

Over the years emulation modelling has developed steadily from its predecessor (simulation) and has been used in different ways in various environment. While majority of the literature relates the success stories of implementing emulation

technology, very few discuss the technical development details of the venture. As such, since emulation is closely related to real time and on-line simulation, some of their applications and development details are also quoted to exemplify the emulation model building.

2.5.1 Automated Material Handling System (AMHS)

Literature search has found that Automated Material Handling System (AMHS) has the most number of applications of emulation. Below are some examples of emulation projects to illustrate the usefulness and benefits of emulation in Automated Material Handling System (AMHS) projects. Included are some technical details on their development and some cautions with regards to emulation model building by the experienced modellers.

Rapistan Conveyor

(LeBaron and K.Thompson 1998) Rapistan Systems, MI, USA and AutoSimulations Inc., UT, U.S.A. used emulation to develop, test, debug, and optimize a complex pick and pack conveyor system for their client. The project integrates a simulation model with the actual control system. The simulation model provides the output for evaluating control logic and algorithms as well as a real time 3-D graphical animation for improved visibility and confidence.

The emulation model they made has a built-in message handler that receives and sends messages through a standard network interface (TCP/IP). The simulation message handler pulls message information from the server at predefined time intervals and acts on these messages. In addition, the simulation model sends messages to the server when certain events have occurred within the simulation model. Emulation has provided them the graphical and statistical output needed to accurately evaluate different algorithms and control logic.

Eskay ASRS

Eskay Corporation (Salt Lake City, UT, USA) an AMHS supplier, developed emulation of a 2-aisle pallet Automated Storage and Retrieval System (AS/RS) Unit Load system to test the order fulfilment control system. The conveyors taking pallets of product to and from the AS/RS were controlled by Think & Do™ industrial control PLC software and the Warehouse Management System (WMS). The simulation model built using Automod was connected to the controller via an OPC server, and to the WMS via sockets.

(Young and Heider 2002) who were involved in the project reported that despite delays created by resource scheduling problems, the emulation was completed months before commissioning was to start. The project manager and software engineers were able to test the full system functionality tested and the checklist over 80% complete before travelling to site

They also attributed that majority of the work for the emulation project involves creating a detailed hardware model and interfacing with the other emulation components. Their work on the interfacing was reduced by using the Automod Model Communication Module (MCM). Nonetheless even though AS/RS, case conveyor, and pallet conveyor are common AutoMod components, for emulation they require significant customization from standard AutoMod operation.

Some important cautions they provided were:
(1) Testing all of the functionality of a MHS would be extremely difficult. It is more important to set limited goals to test and refine basic system functionality.

(2) No amount of software testing can make up for hardware installation difficulties. If the hardware installation is not complete per schedule, the software commissioning will be delayed.

(3) Even if the full system emulation operates correctly, any hardware mistakes will not be revealed until commissioning.

Coca Cola Packer

Coca Cola Enterprise (CCE) Atlanta, GA, USA, needed to accurately model their complex production line and was looking for a method to reduce debugging time on line system start-ups. They developed an emulation model of their Fort Worth Texas production line using Automod software and utilised AutoMod MCM (Model Communication Module) to establish communication between the PLC hardware they use to control their production line and the model (Hodgson and Kartz 2000).

The model was built with a modular structure that supports rapid restructuring to describe different production lines. Photoeye objects, motors, and other resources are modelled in the simulation and the state of these objects are set and read by the PLC connected to the simulation computer using DDE commands provided in the MCM.

By using the model with different speed inputs, different MTTF/MTTR numbers, they made an informed decision that replacing the existing packer with a faster one would not improve the overall line efficiencies after all.

CCE also used emulation to determine the best sequence in which to build a series of 'layered' pallets. By using the 'Dynamic Scheduler' they were able to show an additional 17% gain in throughput (Cheshire and Hodgson 2001).

General Motor Car Assembly

General Motors (GM) used MCM and Automod to do emulation of a GM Holden car assembly plant to validate their After Paint Mix Bank control logic before implementation in the plant. Communication capability was established between AutoMod emulation model and Softlogix 5, an Allen Bradley PLC emulation software using RSlinx, a communication package also from Allen Bradley. The

communication protocol used was Dynamic Data Exchange (DDE) which was provided by AutoSimulations.

In the case where there were more than two models, the Multi Model Synchronization (MMS) Server, an extension of the Model Communications Module was required. The MMS Server automatically opens and manages a connection between each model and the Server and synchronizes all the models that are participating in the simulation.

GM Holden acknowledges that the ability to connect to actual control systems eliminates the need to recreate control logic in simulation models. This not only saves time but also increases model accuracy. GM Holden also realized significant savings using this functionality when their After Paint Mix Bank was fully operational just 3 days after implementation, resulting large financial benefit (Vedapudi 2001)

Schipol Underground Logistic System AGV

(Versteegt and Verbraeck 2002) applied a four-step approach of using simulation in evaluating real-time control systems of Automated Guided Vehicle Systems (AGVS) and Automated Material Handling Systems (AMHS) for the Underground Logistic System (OLS) Schipol in The Netherlands.

The four steps are (1) Testing in a fully simulated environment or offline simulation (Type B in Figure 2.2), (2) Emulation of logistic resources, (3) Combining reality in the loop, emulation and simulation, (4) Implementation of both control and system being controlled in reality.

The strategy in the approach was to solve as many of the technical uncertainties at the first stages and delay the investments in expensive control software and physical logistic resources to later stages. In a fully simulated environment problems can easily and quickly be detected and possible solution can be evaluated for their effectiveness. In later phases the high investments in control software are made, only when the uncertainties and problems are solved. When the uncertainties are solved in

the beginning of the project, the chances of investing in wrong technologies is minimized.

The main idea behind the approach is the development of interchangeable simulated, emulated and prototype components of the control systems and the systems-being controlled. Interchangeable means that components can be changed during experiments without making changes to the control systems.

Some important lessons taken from their work are:

1. The interfaces between the components were defined right at the beginning of the project. Later models and logistic resources had to comply with these interfaces.

2. Use asynchronous messaging to reduce the effect of delays when exchanging information between system components that are coupled in a network.

3. Synchronization between simulation clock and the wall clock is very important aspect in combining simulation, emulation, and prototypes. Except for the software packages that offer standard built-in features for real-time progress in simulation models, separate program has to be made to provide this functionality.

Airport Baggage Handling System

As reported by (Rengelink and Saanen 2002), baggage handling has become one of the major issues in competition between airports. Due to the nature of non-stop operation 24 hours a day, 7 days a week and high security levels, there is a high demand for the quality of the newly implemented systems and its controls at airports. The extensions or changes need to be thoroughly tested in advance without involving the real equipment on site but under conditions comparable to operational conditions.

In view of this, (Rengelink and Saanen 2002) developed a simulation environment for emulating baggage handling equipment which enabled detailed tests and provided insight into the behaviour of the real PLC for their client.

They used eM-Plant simulation environment due to the following advantages:

1. A large number of basic processes were already depicted within the baggage simulation library.

2. The required details for the emulation were able to be easily implemented because of the Object Oriented structure.

3. This approach saved the manufacturer significant lead time in the project and reduced the required time for testing on-site.

In general, the applications of emulation reported above have indicated that in the long-term, simulation and emulation should be used as an integrated part of the design process, analysis, tests and realization. Optimal use should be made of the activities done during all of the phases and this requires reusability of models, easy adjustability for different lay-outs, and project management based on the developments.

2.5.2 Manufacturing Process Control

Although simulation has been the tool of choice for modelling the behaviour of Manufacturing Systems, the accuracy of simulation tools in modelling modern manufacturing systems such as Flexible Manufacturing Systems (FMS) has been doubted. The inherent inaccuracy of simulation tools arises from the inability to model all the constraints associated with the operation of an FMS. Simulation tools usually focus on modelling the primary job entity as it flows through a stochastic

queuing network representation of the FMS. Few simulation tools readily permit the modeller to consider the flow of the supporting resources (e.g. tooling, part kits, fixtures, and processing plans).

FMS Emulator

(Davis et al. 1996) argued that the coordination of all entity flows in an FMS is crucial, and it is the interactions among the controllers within the FMS that coordinate these flows. To address these requirements, they introduced the notion of coordinated object, which included an intelligent controller or coordinator to perform integrated on-line planning and control. They developed a coordination architecture called Recursive Object-Oriented Control Hierarchy (ROOCH) for assembling the coordinated objects into a real-time management structure such that planning and control are both distributed and controlled.

To model the performance of the ROOCH, a simulation methodology, the Hierarchical Object-Oriented Programmable Logic Simulator (HOOPLS) was developed. HOOPLS employs an object-oriented architecture, and the C++ programming language was selected for implementing of the simulation model.

To demonstrate the benefits arising from both the ROOCH architecture and the associated HOOPLS modelling paradigm, (Gonzalez and Davis 1997) developed a physical emulator for an FMS in a laboratory environment at University of Illinois at Urban-Champaign, USA. The development of real time control architecture for a physical emulator is described in (Gonzalez and Davis 1998)

TSCS

(Peters et al. 1996) presented a simulation control system developed by the Texas A&M Computer Aided Manufacturing Laboratory (TAMCAM) to explore the advantages and disadvantages of on-line simulation for process control.

The TAMCAM Simulation Control System (TSCS) consists of a simulation-based controller developed in Arena, a message router and client controls developed in Microsoft Visual C++, and an external database system developed in Microsoft Access. The simulation-based controller is built in Arena using the Arena Real-Time template. The Arena model also uses a user-coded dynamic link library (DLL) written in Microsoft Visual C++ to provide the implementation-specific communications functions required by the router. All connections within the real-time system are implemented using the TCP/IP protocols. The connections within the forecast system are implemented with Database Access Objects (DAO).

2.5.3 Other Applications

2.5.3.1 Control Architecture Evaluation

A modular experimental test bed was developed by (Rogers and Brennan 1997; Brennan 2000) to investigate the relative performance of any variety of manufacturing control architectures with any type of manufacturing system. In order to make this analysis possible, the experimental test bed needs to separate the control system from the emulated system allowing each to be developed and tested independently.

Brennan noted that emulated manufacturing system was chosen over a physical system in order to overcome drawbacks of physical systems for this type of experimental work namely difficulty in controlling the test bed and its environment, and difficulty in reproducibility of tests.

The approach to de-couple the manufacturing system and the control system as fully as possible resulted in the basic structure of the modular experimental test bed as shown in Figure 2.3.

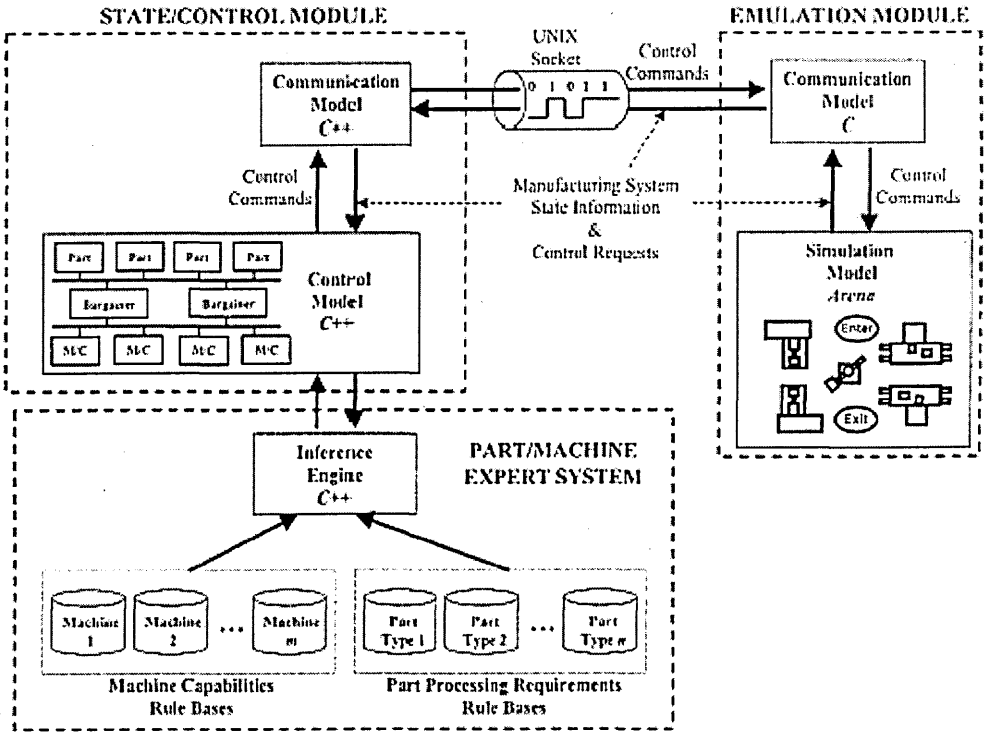


Figure 2.3 The experimental test bed

(Brennan 2000)

This figure shows the two main modules of the test bed, which can be identified as:

(a) an emulation module, which is intended to emulate the behaviour of the manufacturing system being controlled, and

(b) a state/control module, which is used to implement alternative decision-making schemes.

The simulation model was written in Arena simulation package, and can be modelled relatively easily to represent alternative manufacturing system configurations. The Arena simulation model is augmented by a communication model, implemented with additional ANSI C routines, which carries out the low-level communication functions via input/output streams (implemented as UNIX or INET sockets).

The state/control module, which is used to implement the test control architectures, is implemented in the C++ programming language. The structure of this module is similar to the emulation module; it consists of a communication model, used to carry out the low level communication functions, and a control model, used to implement the test control architectures.

A part/machine expert system is included as part of the state/control module to instantiate the control system decision makers. The expert system relies on a set of rule bases that describe the capabilities of the work centres in a given manufacturing system and the detailed processing requirements for the parts that are to be introduced to the system (Brennan and Norrie 2001).

2.5.3.2 Verification of Controller Software

Testing the behaviour of a controller for example a PLC, which controls a device being part of a more complex system, is usually done by connecting the controller to a 'stand-alone' version of the device called 'mock-up'. This method of verifying and validating the controller's software is expensive, and test conditions are hard to reproduce. Such tests are incomplete since the interaction of this device with the other parts of the system is simply ignored. Therefore a large part of testing and debugging is still carried out on-site.

To solve such problems (Schludermann et al. 2000) has developed a Soft-Commissioning (SoftCom) , a 'hardware-in-the-loop' (HIL)-based system approach that enables interaction between controller such as a PLC and a commercial discrete event simulator, also known as emulator. HIL means that the inputs and outputs of a controller are connected to a simulation (emulation) of the part to be controlled. Hence, the system needs to be modular and scalable.

While most other HIL systems are based on continuous real-time simulation and use fast Digital Signal Processor (DSP), SoftCom was developed to interact with commercial Discrete Event Simulators (DES) and conventional I/O hardware. A

special communication protocol was defined to make flexible SoftCom internal data exchange possible. The two basic modules shown in figure 2.4 are the I/O Devices Driver (IODD), which is used to interface between the I/O hardware and the SoftCom protocol, and the Simulator to real World Interface (SWI) which is used to link the simulator to the SoftCom system.

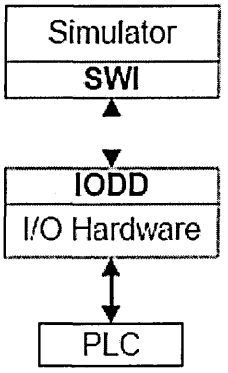


Figure 2.4 The SoftCom System

The IODD internal link to the I/O cards is defined by a library interface, e. g. a Dynamically Linked Library (DLL) in the Windows world. The implementation of this interface depends on the I/O hardware in use. Thus the IODD must support connections to more than one library at the same time to be able to establish links to different I/O cards.

The implementation of the SWI depends on the simulator's approach of providing access to its variables and objects. (Schludermann et al. 2000) prototype was based on the simulation environment Arena. Arena provides two mechanisms for external programs to interact with the simulation, Visual Basic for Applications (VBA) module and Dynamically Linked Library (DLL) interface. DLL interface was chosen as the link between the SWI and the simulator for the reason for that the DLL can be implement in C++, which provides more flexibility in programming.

The DLL interface defines routines to interact with the simulation: One type gives access to simulator variables and the event calendar. The other type enables the

simulation to execute user code when appropriate events are triggered. By forcing the simulation to call into the user event DLL routine periodically, a certain update time step is achieved. This time step is needed to synchronize both simulation data and simulation time with the SoftCom system.

Tasks such as 'system update', 'system configuration' and 'runtime control' are usually quite simple as long as the system is restricted to a single computer. But they rapidly grow in complexity with the number of computers involved. Thus configuration and maintenance of bigger systems can become a time-consuming job.

The strategy to simplify this job is to update and configure the system on a single computer, which then forwards this information to all other computers of the system. Based on this strategy, a SoftCom Manager was developed as a centralizing tool for configuration and runtime control (Schludermann et al. 2000).

2.5.3.3 Simulation Model Integration

One of the greatest challenge towards building emulation model is having to integrate different functional areas developed in different simulation environment as reported by (Boer et al. 2002) .

In a complex system, such as a container port, there are thousands of pieces of equipment and controllers. Testing of complex systems like a port system might entail several difficulties, which, beside the general communication problems, concern the variety of simulation environments, variety in the real equipment and differences between communication protocols.

The simulation models or simulation components are usually developed by different modellers, using different concepts and different simulation environments. Thus, the communication between various environments should be enabled in order to provide collaboration. Equipment and simulation models support different communication protocols, therefore, different models can communicate only if a common protocol is

worked out or several interfaces are developed, that allows for communication between any two of them.

In view of this, (Boer et al. 2002) has developed FAMAS (First All Modes All Sizes) Simulation Backbone Architecture, a flexible architecture for the interoperability between various distributed simulation models. Its structure shown in Figure 2.5 consists of technical components representing the simulation models, and technical components providing common tasks used by the functional components.

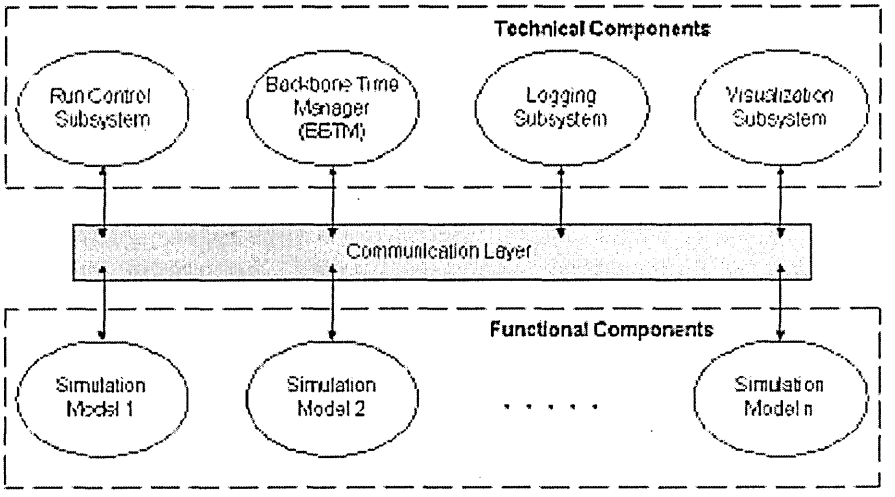


Figure 2.5 The Structure of the FAMAS Simulation Backbone Architecture

In the development of the FAMAS Backbone Architecture (Boer et al. 2002) listed requirements to be fulfilled, which are:

Distributed execution: this can be achieved by a well-defined interoperability between different simulation components. The interoperability in the FAMAS Simulation Backbone is provided by a low-level message passing mechanism.

Optimal communication: effort is required to attain an effective communication speed.

Stand-alone and distributed testing: refers to the possibility to test distributed simulation models developed by different parties as in standalone as in distributed environment.

Package independence: this requirement focuses on combining simulation models implemented in different simulation packages (e.g. Arena, eMPlant, Enterprise Dynamics) and programming languages (C++, Java, Delphi, etc.). The characteristics mentioned so far reflect the grade of flexibility of the architecture and reusability of the simulation models.

Structure transparency: aims to give some insights into the architecture for the groups who intend to develop models or support subsystems for it, in order to provide interoperability. The transparency helps the modeller to couple the simulation models effortlessly.

Hierarchical structure allows for modelling, design, and development in a hierarchical manner. This feature is essential in the FAMAS project as the models might be developed at different levels of detail.

Some other application areas that share some common elements of emulation model include the following.

- on-line business process and decision making (Dalal et al. 2003),
- real time security control system testing (Jordan et al. 1998; Smith et al. 1999),
- transport (Verbraeck and Versteegt 2000; Verbraeck and Versteegt 2001; Hunter and Machemehl 2003; Xu et al. 2003).

These were also reviewed in the development of questionnaire survey described in Chapter 3.

2.6 Summary

The literature search was based on the pretext of trying to understand a simulation technology in manufacturing system design called emulation. As it progressed, it was discovered that while there were much interest on emulation, not many companies were willing to invest in the technology, primarily due to the monolithic nature of the model building and high cost. Thus a new approach towards developing emulation model needs to be investigated including the possibility of simulation-emulation conversion process.

Review on the similarities and differences between simulation and emulation models have highlighted several issues to consider before developing a new method for building emulation model. They include (1) simulation and emulation models are built with different aims, (2) emulation operates with hardware in the loop (HIL) which requires an interface for the interaction and has to be run in real time, (3) emulation models higher level of detail compared to simulation model, (4) control rules are hard coded in simulation where as control rules are separated from event code in emulation, and (5) runs are repeatable and results are predictable for simulation but not for emulation.

Review on the application areas of emulation has shown that most of the applications are in manufacturing area particularly in the modelling of automated material handling system (AMHS). Others include manufacturing process control, transportation logistics and non manufacturing application.

With regard to modelling work, majority of the work reported for the emulation project involves creating a detailed hardware model and interfacing with the other emulation components. While Object Oriented approach, the likes of HOOPLS and eM-Plant simulation environment offers the advantage of flexibility and openness, most manufacturing application simulation software packages (for example Arena, Automod, Quest etc) are non object oriented. However there are functions or modules created either directly by the vendor or indirectly by the user of simulation packages to overcome drawbacks encountered in the respective projects.

An important observation on the existing method of building emulation model was that it had to be developed separately from simulation model, in other words they were not convertible. To make emulation model more cost effective, as indicated from the result of the literature review, a hybrid simulation-emulation model (HSEM) is seen to be a viable alternative.

The next chapter describes the overall methodology by which the research was carried out including developing areas identified in this chapter.

CHAPTER 3

JUSTIFICATION OF THE PROPOSED RESEARCH AND RESEARCH METHODOLOGY

3.1 Introduction

This chapter describes the justification of the proposed research and the methodology employed to answer the issues raised in the previous chapter with regard to developing new methodology of building emulation model.

The first part consisting of preliminary research and questionnaire survey describes justification process of the proposed research. The second part consisting of development of a sample model, development of a new methodology to develop emulation model and validation describes the research methodology of the proposed work.

3.2 Preliminary Research

Initial research works involved fact finding through literature review and gathering of information from simulation software suppliers and users. Two important initial findings were:

- The lack of published information concerning the procedure of emulation model building,
- Emulation is not yet widely used especially outside USA.

Conference papers especially from the annual Winter Simulation Conferences provided vast information on the benefits and applications of emulation. However, not much information could be obtained with regard to its building methodology. As also noted in 2.2, further search was done on the websites and journals using the phrases 'simulation for control system testing' 'real time simulation' , as well as 'flexibility of adding details' which provided considerable amount of related material.

The initial literature findings coupled the outcome of meetings and correspondence with simulation software suppliers and users provided a clearer meaning of emulation the context of the proposed research.

To identify current trend and expectation research priority areas within the proposed research, a questionnaire survey was developed and conducted.

3.3 Questionnaire Survey

The main objectives of the survey were to investigate the extent of use of Emulation model for Control System Testing, its application areas and users opinion about its model building.

Due to the similarity in purpose and more commonly understood term, "simulation model for control system testing" instead of "emulation" was used in the questionnaire survey.

It was also discovered that the phrases have also been used in non manufacturing application areas like security system (Jordan et al. 1998; Smith et al. 1999) and

transport system (Verbraeck and Versteegt 2000; Hunter and Machemehl 2003; Xu et al. 2003).So, for the purpose of initial study and comparison the scope of investigation was widened to include non manufacturing application areas.

The questionnaire questions were developed based on guidelines in (Oppenheim 1992; Thomas 1999). Regarded as the biggest world wide gathering of simulation practitioners and simulation software vendors, Winter Simulation Conference (WSC) was chosen as an ideal opportunity to distribute the questionnaire and gather the required information.

The questionnaire distributed to the participants of WSC 2003 consisted of seven topics dealing with

1) type of user ,
2) whether or not using simulation for control system testing,
3) current application area,
4) ranking the benefits of using simulation for control system testing,
5) important stages for its model building,
6) simulation packages used
7) potential application areas.

The first two general topics were used to classify the background of the respondents according to their professional affiliation and to gauge the level of use of simulation in control system testing in general.

Topics (3), (4) and (5) only apply to current users of simulation for control system testing. The aim was to gather information based on their experience the benefits of emulation model and facilities that would assist the development of emulation model.

The last two general topics deal the choice of simulation software used and the simulation user's expectation on the future use of emulation. Apart from indicating the preference towards certain software package, more importantly the response to the

choice of simulation used would identify the suitability of the software to be used for control system testing from the user's perspective.

With regard to the application area, present and future, the respondent was offered choice(s) from a list of which the general meaning of each application area was as follows:

Process control involves monitoring, controlling and improving a process typically in production environment.(Davis et al. 1996; LeBaron and Hendrickson 2001)

Business Process is collection of activities designed to produce a specific output for a particular customer or market. (Aguilar-Savén 2004)

Material Handling is the movement, storage, control and protection of materials, goods and products throughout the process of manufacturing, distribution, consumption and disposal. (Mueller 2001; Versteegt and Verbraeck 2002)

Security system is the mechanism to protect facilities against intrusions by external threats as well as unauthorized acts by insiders. It includes physical as well as information protection.(Jordan et al. 1998; Smith et al. 1999)

Transport system is the facility consisting of the roads and equipment necessary for the movement of passengers or goods. Mode of transport includes land, air and water.(Verbraeck and Versteegt 2000; Verbraeck and Versteegt 2001)

As to the benefits of using simulation for control system testing, the respondent is asked to rank them from a list. They are (i) shorter commissioning time, (ii) low overall cost, (iii) efficient use of resource and (iv) client satisfaction.

The respondent was also asked in topic (5) to rank according to its importance the development stage requiring specific tool for emulation model building. The development stages gathered during initial case studies are (i) Interfacing between models, (ii) Modifying simulation code, (iii) Determining the correct level of detail.

The response would help to indicate the inhibiting factors at present as well as the important areas of research and development.

The questionnaire also contained additional space so that respondents could specify particular application areas, benefits and simulation packages that were not listed in the original choice of answers. This was aimed to get more information that was not retrieved during the literature review exercise.

The survey sample was not selected by any formal statistical method. Out of 100 samples distributed, answers from 26 respondents were collected and analysed. Table 3.1 shows the distribution of current and potential areas using emulation for control system testing based on the user background. Table 3.2 shows the distribution of the application area of emulation, present and future from simulation practitioner's perspective.

Table 3.1 Current and potential areas using of emulation for control system testing

		User Type			
		Academic (%)		Industrial (%)	
		Current	Potential	Current	Potential
Application area using emulation for control system testing	Process Control	28.6	27.6	33.3	33.3
	Business Process	28.6	17.2	14.3	18.2
	Material Handling	28.6	20.7	28.6	24.2
	Security System	0	10.3	0	3.0
	Transport System	14.3	24.1	23.8	21.2

Table 3.2 Present and Future Application Area of Emulation

	Present (%)	Expected (%)
Process Control	33.3	30.9
Business Process	18.5	16.4
Material Handling	29.6	23.6
Security System	0	7.3
Transport System	18.5	21.8

The results of ranking the benefits of using emulation for control system testing based on weighted average calculation were as follows:

(1) Efficient use of resource

(2) Low overall cost

(3) Shorter commissioning time

(4) Client satisfaction

Ranking of the importance of development stage requiring specific tool for emulation model building would indicate (1) inhibiting factors at present, (2) important areas of research and development. The results of survey, in order of importance, are as follows:

(1) Interfacing between models,

(2) Modifying simulation code,

(3) Determining the correct level of detail.

Based on the preliminary research and analysis of questionnaire survey, several important points were identified. They are listed below.

- Process control and material handling are considered to be the prime application areas of emulation.
- Emulation has the potential being used for control system testing in areas other than manufacturing and production like transport system, business process and security system.
- There is a need to provide a generic methodology and facilities for developing emulation models.

3.4 Case Studies

The case studies that followed were conducted to look into:

(1) Defining the requirement specification for a simulation model to be able to convert into an emulation model.

(2) Developing a framework of activities associated with building a simulation model and converting it into an emulation model.

Results of the case studies from the first part were used to develop the requirement specification as discussed in Chapter 4. They include adaptability for modelling detail; inter process communication and criteria for simulation/emulation convertible software.

Case studies for the second part looks at detail simulation model development procedure as well as interfacing real system with the simulation model. This is discussed in the development of proposed framework of activities in Chapter 5.

3.5 Development of a sample model

Based on the more in-depth case studies, a methodology of converting simulation model into emulation model was developed. The methodology consists of three main steps. They are as follows.

1. Developing conceptual model,
2. Converting an initial or conceptual simulation model into a more detailed simulation model,
3. Integrating the model at various level of detail with external controller and running it in real time.

The methodology was verified by developing, constructed through the various stages, a sample model using Arena Simulation package.
The reasons for choosing Arena simulation package were as follows.

- It is a general purpose, non specific application software with which a generic tool could be developed for a wider use in a variety application area,
- It is modular, hierarchical and configurable which would allow integration of models of various level of detail.

- It has a real time modelling facility which would allow real time communication between the simulation model and controller.

The case example is the modelling of a 2-machine manufacturing system. Two types of parts namely "Part Type 1" and "Part Type 2" are processed in the system. A "Part Type 1" is processed first on Machine 1, then on Machine 2. A "Part Type 2" is processed first on Machine 2, then on Machine 1.

The sequence of events the entities, Part Type 1 and Part Type 2, follow is shown in Figure 3.1

The Arena model, in Figure 3.2 shows the modelling logic and animation of the manufacturing system based on the assumed time projection of events. In reality assumed time projections are not enough. For example, due to limited buffer space the machines need to communicate between each other as well as with the arrival controller to monitor the number of parts coming to the respective stations.

The model, shown in Figure 3.3, is a modification of the simulation model incorporating Real Time elements. Only the logic diagram is shown as the animation diagram in this example is exactly the same as for simulation. This model demonstrates Arena running in execution mode and conducting inter-process communications with an external client application called RTConsole.exe written in Visual Basic.

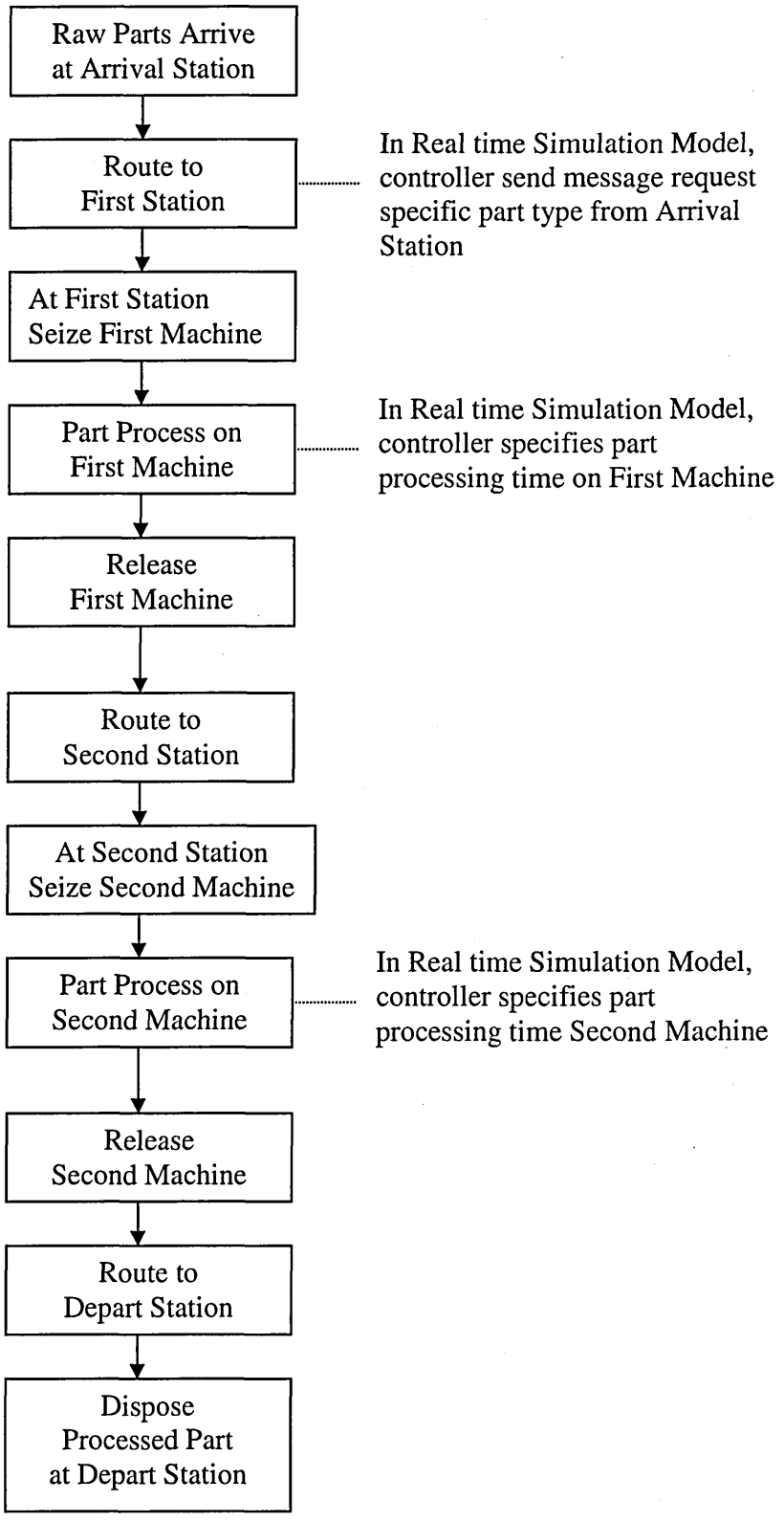
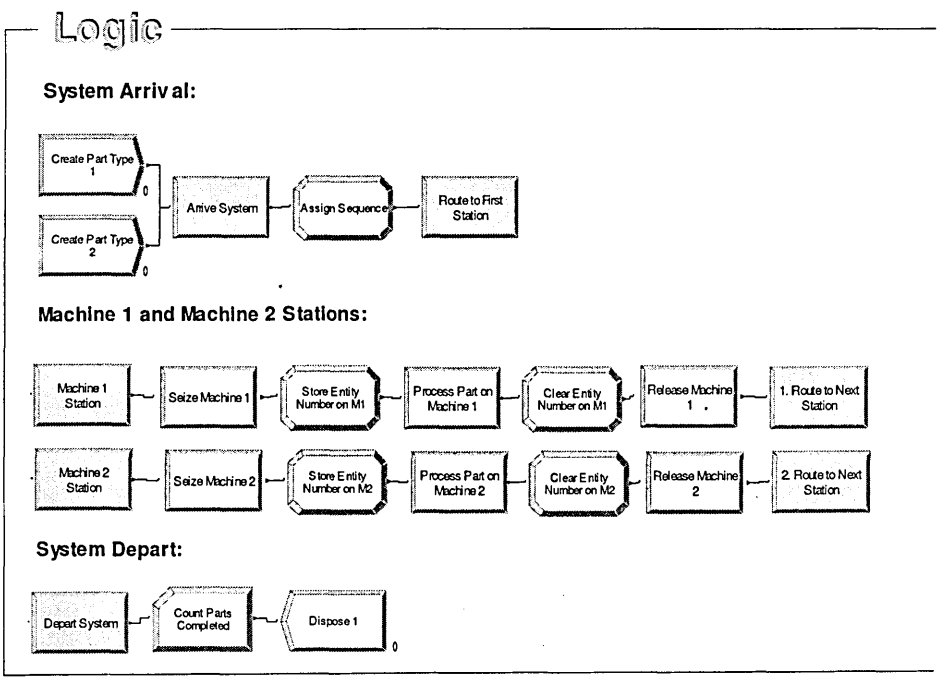


Figure 3.1 Sequences of Events for the 2-Machine Manufacturing System

Simulation



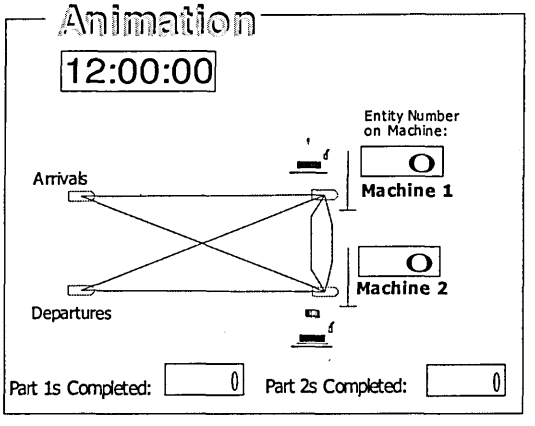


Figure 3.2 Simulation of a 2-Machine Manufacturing System

Emulation

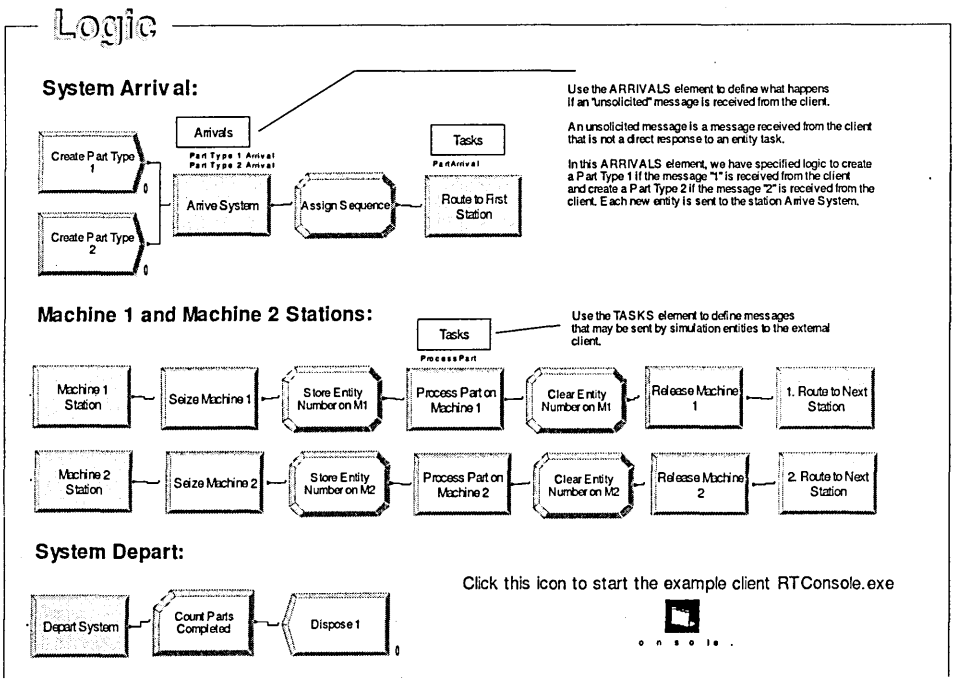


Figure 3.3 Real-Time Simulation of a 2-Machine Manufacturing System

The conversion involves some changes in the communication structure as well as the message handling in the simulation model.

The simulation of sending and receiving of the messages and relevant parts by the respective machine on a 'client' was successfully controlled through a remote computer, which acted as a 'server'.

Further information on the development of a similar sample model can be found in (Hasnan and Perera 2004).

3.6 Development of new methodology

Based on previous studies and experience of developing a sample model, a new methodology of building emulation model is developed.

The development comprises of the following activities.

1. Defining the requirement specification for a simulation model to be able to convert into an emulation model.
2. Developing a framework of activities associated with building a simulation model and converting it into an emulation model.
3. Documenting the methodology for general application.

The specific requirements for hybrid simulation-emulation model (HSEM) building are described in Chapter 4. The methodology for the development is documented in Chapter 5.

3.7 Validation

The validation of the new methodology is through the development of a hypothetical simulation model of a small manufacturing plant and converting it into an emulation model where some form of external control is included. The development of the validation model, based on the general methodology developed, comprises of the following activities.

1. Building base simulation model.
2. Modelling details.
3. Integration with controller.

The validation process is described in Chapter 6.

3.8 Summary

This chapter has described the justification process of the proposed research as well as outlined the methodology of the research. The stages, sub-activities and areas as well as the outcome at each stage of research are summarised and shown in Figure 3.4.

The next chapter discusses the requirement specifications for developing HSEM as prescribed from stages 1, 2 and 3 of the research.

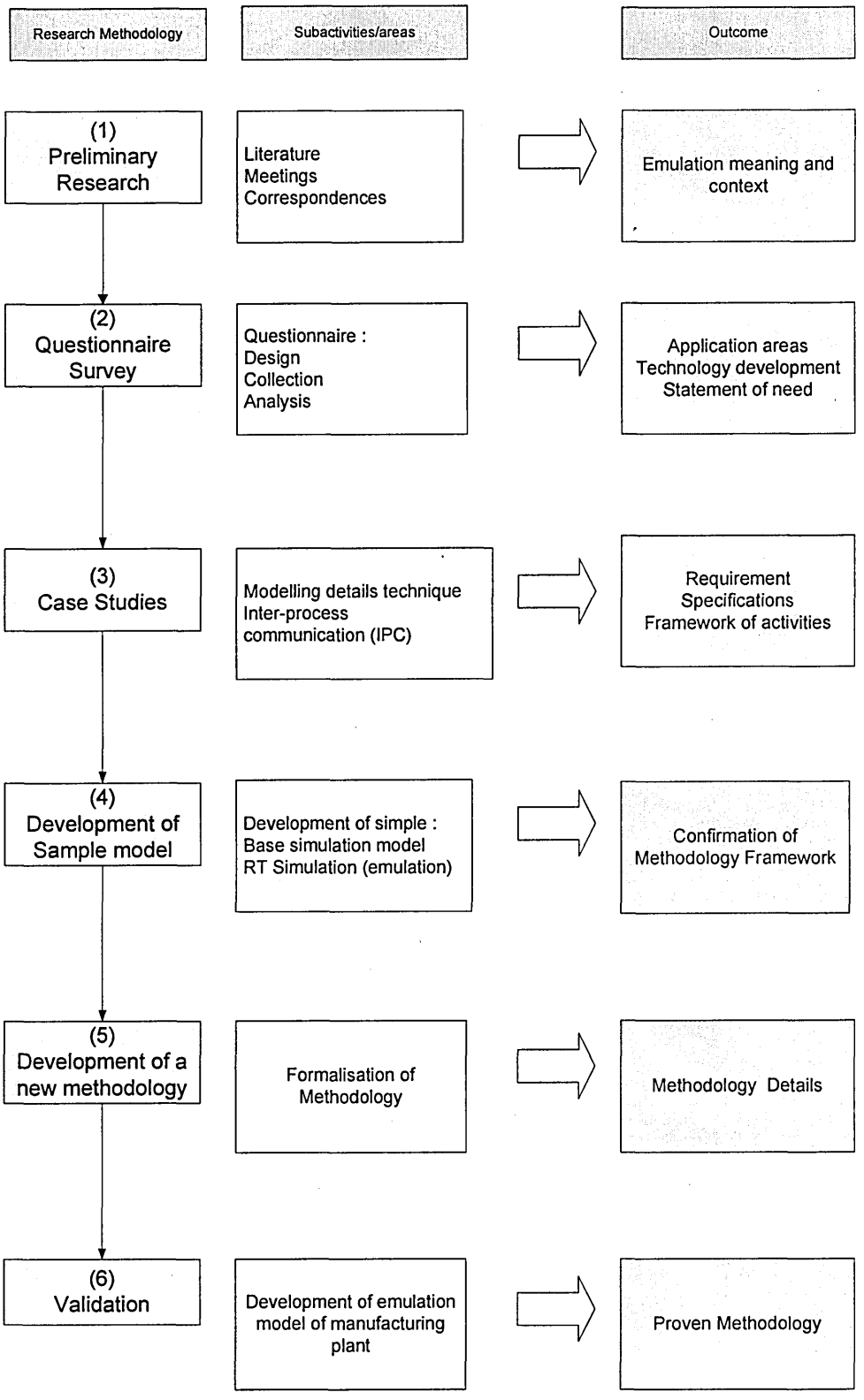


Figure 3.4 Research Methodology Summary

CHAPTER 4

REQUIREMENT SPECIFICATION FOR SIMULATION-EMULATION MODEL BUILDING

4.1 Introduction

The comparison of salient features between simulation and emulation model summarised in Table 2.1 in Chapter 2 has highlighted the general requirements towards building emulation model from existing simulation model.

This chapter discuss in more detail the technical requirements for the possible conversion. It covers two major categories, namely (1) the flexibility of adding details to the simulation model while assuring its correctness and (2) the inter process communication between model and real control system. It is followed by discussion on selection criteria of simulation software suitable for the development of Hybrid Simulation/Emulation Model (HSEM). This chapter concludes with a summary of the requirement specification for HSEM building.

4.2 Flexibility for Modelling Detail

Like any system that anticipates change in the later stage, HSEM needs to be build with flexibility in mind. One of the most important concerns about the development of HSEM is the requirement of adding components with multiple levels of detail or abstraction as the model develops. Simulation like any modelling project begins with a conceptual model with many assumptions. It then goes through an iterative process becoming more detail and refined until it reaches a stage considered accurately representing the real system, a process shown in Figure 4.1.

As suggested by(Robertson and Perera 2002), due to this iterative fashion of model building it is considered to be a good practice to embed the first version or iteration of the model with flexibility. If this first iteration of the model is designed to accommodate such predicted developments, the model will require less time and effort later in the model development life cycle, for the 2nd, 3rd, 4th, or even 5th iteration. The consequences of not embedding this flexibility initially are that the 2nd iteration will require greater time and effort to be modified, as will the 3rd iteration, and so on.

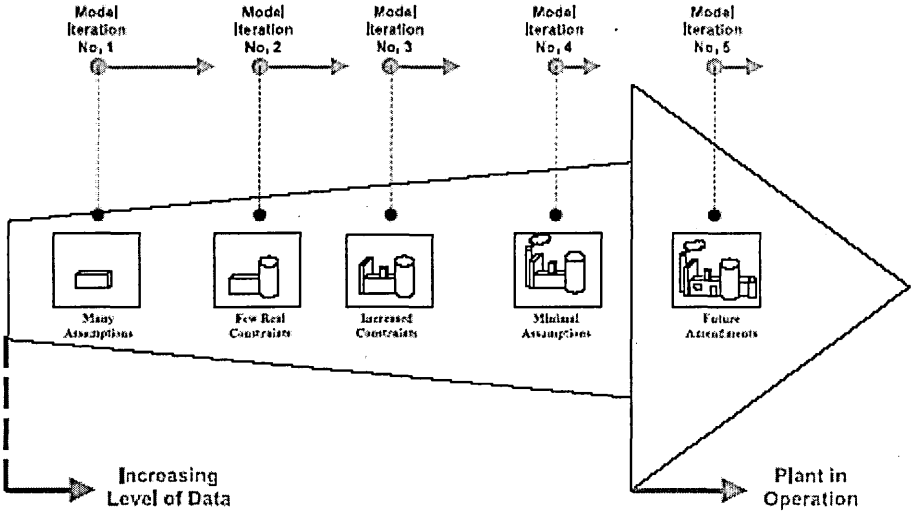


Figure 4.1 The iterative model building process
(Robertson and Perera 2002)

The degree of **flexibility** should be sufficient to allow the user to create a "hybrid" model where the control system to be verified provides the logic for a part of the model and the simulation product takes care of the rest.

4.2.1 Adaptable Simulation Models

Since building a simulation model can be a difficult and time-consuming task, a decision-maker will seek to reuse a simulation model if possible and change it to solve a different problem or evaluate another option. Thus, it is desirable to have adaptable simulation models that are easy to change with little or no programming effort (Randell et al. 1999).

In the manufacturing systems context, small changes in the manufacturing environment can produce many different, though related, changes to the data input for the simulation model. Some examples of changes that are likely to occur are: (a) the answers needed from the simulation, (b) the products that are being made on the shop floor, (c) new production processes or characteristics of the current production processes, and (d) changes to the plant layout.

Also, the process of designing a manufacturing system requires changes to the simulation model. As a manufacturing system progresses from a concept to a detailed design to an installed and operating facility, the simulation model of the system must change. Typical changes include equipment selection and location, control rules and operating procedures for equipment and material handling systems, arriving material and customer order characteristics, and operating hours.

4.2.2 Modular Simulation

Simulation experiments often require the examination of a potentially large number of scenarios dealing with many solution strategies. The development time to build new models or make changes to existing models can be quite substantial and

problematic. One technique to reduce this problem is to develop generic, modular simulation solution systems.

Modularity is based on locality and encapsulation (Pidd and Castro 1998). Locality is the notion that all information relevant to a design decision should be kept in one place, i.e. within a module. Encapsulation, or information hiding, separates internal, hidden aspects of a module from the external (Rumbaugh 1991).

A modular model should satisfy two conditions according to (Pidd and Castro 1998):
(i) The model or component must not directly access the state of any other model or component.
(ii) The model must have recognized input and output ports through which all interaction with the exterior is mediated.

A further advantage is that this modularity supports the re-use of model components, since modular models are defined with no direct reference to the state of their potential co-components. This improves the likelihood that modular models may be built, at least in part, from existing components. It also provides an attractive way of introducing hierarchical components into simulation models (Robinson et al. 2004).

The general modelling approach that is recommended is to remain at the highest level possible when creating the models. However, as soon as we find that these high-level constructs do not allow capturing the necessary detail, drop down to the next level for some parts of the model rather than sacrifice the accuracy of the simulation model. (Meinert et al. 1999; Nketsa and Valette 2001)

Some simulation packages seem well suited for modular development while others have a structure that makes modular design difficult. It very much depends on the modelling style that usually follows the programming style which could be either procedural or object oriented.

Procedural Style

Programming in simulation languages like GPSS, SLAM and SIMAN are considered to be procedural based. In the procedural programming, a problem was decomposed into procedures and either represented by general components, like a queue, or represented in programming code with a data structure and code.

(Joines and Roberts 1999)argued that the main limitation of the procedural style was its lack of extensibility. From the earliest simulation languages until the early 1990s, the only way to adapt these simulations was through functional extension. In other words, structural functionality can be added to the simulation but cannot alter any of its basic processes, like giving properties to resources. For example, the simulation needed to include a bridge crane; it has to be completely programmed. One of the reasons for this lack of extensibility was that procedural changes were the only approach to model changes.

Because many simulation languages offer pre-specified functionality produced in another language, the user cannot access the internal function of the language. Instead, only the vendor can modify the internal functionality. Also, users have only limited opportunity to extend an existing language feature.

Object (component) Style

With object style, a simulation language provides a user with a set of pre-defined object classes (i.e., resources, activities, etc.) from which the simulation modeler can create needed objects or components. The modeler declares objects and specifies their behavior through the parameters available. Therefore, an object can be described by an entity that holds both the descriptive attributes of the object as well as defines its behavior.

The class concept evolved out of the notion of encapsulation where objects needed independence of action and a means to hide their implementation details, yet provide

an interface for their use. Further, there needed to be way to construct objects and to communicate among them.

In the context of extensibility, object style can be divided into two types namely object-based and object-oriented.

The **object-based** approach only allows extensibility in the form of composition where new objects can only be created out of existing objects. Object-based programming is a limited version of object oriented programming where one or more of the following applies: (1) there is no implicit inheritance; (2) There is no polymorphism, (3) only a much reduced subset of the available values is objects, typically the GUI components.

Simulation packages like Arena and AweSim have beginnings of object-based. Both languages provide a composition approach to creating network macros, through Arena templates and AweSim subnetworks. Both have access to Visual Basic, which is only object-based. AweSim wraps its functionality in a few objects, whereas Arena contains an object model (not with SIMAN features) that is integrated with Visual Basic. These templates or subnetworks provided a form of encapsulation but these collections do not provide for autonomous objects.

An **object-oriented** simulation (OOS) deals directly with the limitation of extensibility by permitting full data abstraction. Object-oriented programming is a type of programming in which programmers define not only the data type of a data structure, but also the types of operations (functions) that can be applied to the data structure. In this way, the data structure becomes an object that includes both data and functions. In addition, programmers can create relationships between one object and another. For example, objects can inherit characteristics from other object.

The Smalltalk environment is fully O-O and contains fully OOS. Simulation languages based on C++, like C++/CSIM and C++SIM, possess all the object-oriented capability. Simple++ and MODSIM III are further examples of object-

oriented languages that employ most of these concepts within different simulation frameworks (Joines and Roberts 1999).

It has to be noted that object-oriented programming and simulation requires more skill from the user which is the main drawback in its use. Comparison on types of modularity used in some simulation packages are shown in Table 4.1.

4.3 Requirements for inter process communication

The term inter process communication (IPC) describes the act of two applications communicating and sharing data with one another. This feature allows the integration of external data and applications into and out of the simulation models. At the emulation stage, the communication between the simulation model and the real controller has to be established and synchronized.

4.3.1 Real time capability

Real time simulation is based on the ability of the system to obtain the real-time data needed to update the simulation model.

Majority of control systems are designed to operate in real time, so emulation experiments should be operated in real time. Although simulation models can provide responses faster than real time, this is potentially a source of error, as control system timers cannot adapt to this, running at speeds greater than real time should be avoided.

The synchronization of time which is aimed at synchronizing the simulation clock of the simulated control system to the internal clocks of the emulation model is important. The usual implementation of real-time or "wall clock" synchronization is to jump to the next event on the event list, to check whether the time of this event is such that it can be allowed to take place, and if not, delay the simulation environment until the event is allowed to take place. The problem here is that *external* events can

come in before the next event time, while the simulation clock has already been advanced to that next time.

Some software packages offer standard features for real-time time progress in simulation models. Arena, for example, offers a set of extension called Arena RT that allows communication with external clients such as Manufacturing Execution Software (MES), Management support tools and interactive training interfaces. Arena RT also allows enhanced control logic where it can initiate and react to external actions as well as synchronization with external clock. Object oriented simulation software like Simple++ (now eM-Plant) and FlexSim offer standard built-in features for real-time progress as well.

The simulation software selection criteria and comparison in section 4.4 lists the simulation software packages that provide this facility.

4.3.2 Model Communication Interface

The integration of simulation models is based on three fundamental themes: (1) models are objects; (2) they communicate with one another in client/server relationships by passing messages; and (3) each model is represented by an agent that explains the capabilities of the model and assists with integration of that model.

One of the important aspects of emulation is the communication between models. To make real components and simulation modules fully pluggable against each other the simulation models have to provide interfaces similar to the interfaces of the real world plant. Ideally an input or output in the simulation model may "directly" be attached to the output of a sensor or the input of an actuator.

Types of interfaces that can be used are for instance DDE (Dynamic Data Exchange), DLL (Dynamic Link Library), TCP/IP socket connections, ActiveX, OPC (OLE for

Process Control) and DCOM (Distributed Components Object Model). When needed the user should also be able to construct custom made interfaces.

Client-Server Model

A standard model for emulation applications is the client-server model. A server is a process that is waiting to be contacted by a client process so that the server can do something for the client. A typical scenario is as follows:

The server process is started on some computer system. It initializes itself, and then goes to sleep waiting for a client process to contact it requesting some service.

A client process is started, either on the same system or on another system that is connected to the server's system with a network. The client process sends a request across the network to the server requesting a service of some form.

When the server process has finished providing its service to the client, the server goes back to sleep and wait for the next client request to arrive.

In the context emulation for control system testing, the emulation model is regarded as the client while the controller is regarded as the server.

Socket

A socket is a software object that connects an application to a network protocol. A program can send and receive TCP/IP messages by opening a socket and reading and writing data to and from the socket. This simplifies program development because the programmer need only worry about manipulating the socket and can rely on the operating system to actually transport messages across the network correctly.

A socket is one end of a two-way connection between running programs across a network. Generally, the server runs on a machine of which the IP address is known to

the clients, and waits listening for a client to make a connection request. The connection request is made to a specific port number that the server is listening to. That is, in establishing a connection, a client needs to make a request to the server's machine and port.

If the request is successful, the connection is established through a new socket bound to a different port on the server. A new port is required so that the server can continue listening to the original port for new clients. The socket connection is as shown in Figure 4.2.

If a connection is established the client and server can communicate by writing to or reading from the sockets created by the connection.

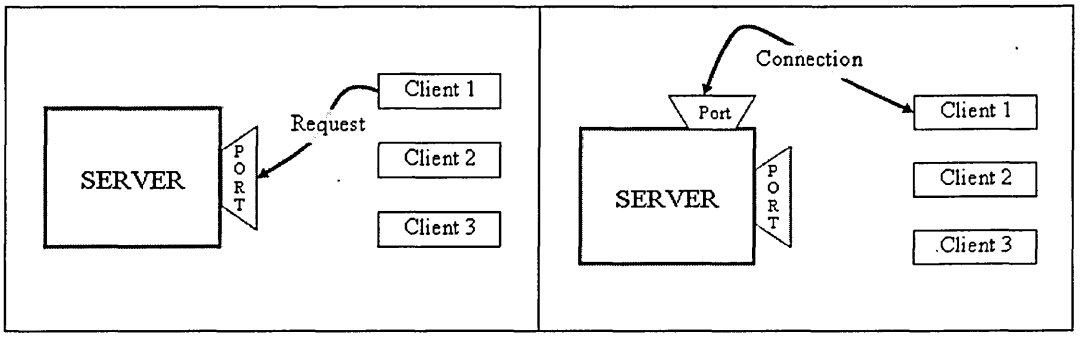


Figure 4.2 Socket Connection

With the Sockets technology such a stable simultaneous work of several Internet applications is available. The socket implementation for MS Windows called Windows Socket or just Winsock.

An important point to note about socket is that the application can produce as many sockets as it needs for effective job, but one socket works with one TCP/IP port only.

OPC (*OLE for Process Control)*

In a multi client/server situation, where interoperability among multiple vendor products has become a problem, OPC could be the solution.

The background of the problem then was the absence of any standard. All process-control and information systems on the market have proprietary techniques, interfaces, and APIs (Application Programming Interfaces) in order to access the information that they contain. The cost of integrating the different systems and the long-term maintenance and support of an integrated environment can be significant.

To overcome this problem OPC (OLE for Process Control) was developed by the OPC Foundation to provide a common interface for communicating in real time with diverse process-control devices, regardless of the controlling software or devices in the process. By using a standard way of configuring computer hardware (and software interfaces) automatically, a device will easily connect to another and immediately work without the need for lengthy installation procedures or complex configuration.

Based on Microsoft's OLE (now ActiveX), COM (component object model) and DCOM (distributed component object model) technologies, OPC consists of a standard set of interfaces, properties, and methods for use in process-control and manufacturing-automation applications. The ActiveX/COM technologies define how individual software components can interact and share data.

Although OPC is primarily designed for accessing data from a networked server, OPC interfaces can be used in many places within an application. At the lowest level they can get raw data from the physical devices into a SCADA (Supervisory Control And Data Acquisition) or DCS (Digital Control System) or from the SCADA or DCS system into the application. The architecture and design makes it possible to construct an OPC Server which allows a client application to access data from many OPC Servers provided by many different OPC vendors running on different nodes via a single object. Figure 4.3 shows the OPC Client/Server Relationship where "Application" could be the emulation model. [I/F refers to Interface, I/O refers to Input Output]

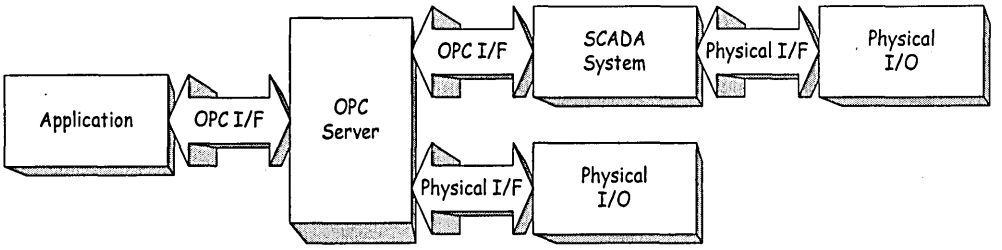


Figure 4.3 OPC Client/Server Relationships
(OPC Taskforce 1998)

Typically, devices react to state changes as directed by the controller; for example a controller might send a message containing values that cause a motor to change its section velocity or stop or start. Devices send messages to the controller to indicate the status of the device, configuration parameters, and so on.

To emulate the devices in the manufacturing system, an emulation model connects to one or more OPC servers as a client application. OPC servers read and write values to a controller in the same way as system devices.

OPC has attracted interest among software vendors and simulation software developers. Arena, for example, has added new enhancements to include the ability to use OPC technology to test control system logic on a model of a manufacturing line rather than testing on the real factory (Bapat and Sturrock 2003).

The Model Communications module (MCM) which is an enhancement to the AutoMod software that allows a model to communicate over a network with other software applications is reported to also support communication with OPC server and sockets (Rohrer and McGregor 2002).

4.4 Simulation Software Selection Criteria and Comparison

This section establishes the important emulation modelling features needed for HSEM project and reviews the readiness of some existing simulation software packages in providing those features.

Emulation Modelling Features

Manufacturing simulation models can be developed using both general-purpose and manufacturing oriented software. General purpose software can solve almost any discrete event simulation problem. Examples of general-purpose simulation packages are Arena, AweSim, Extend, GPSS/H, Micro Saint, MODSIM III, SIMPLE++, SIMUL8, SLX, and Taylor Enterprise Dynamics Developer.

However, depending on the complexity of the system being modelled, manufacturing simulation package can significantly simplify and quicken the modelling process. Examples of manufacturing -oriented simulators are Arena Packaging Edition, AutoMod, AutoSched, Extend + MFG, ProModel, QUEST, Taylor Enterprise Dynamics Logistics Suite, and WITNESS.

Each manufacturing-oriented simulation packages on the market has its strengths and weaknesses. Some packages focus on ease of use and compromise flexibility, while others focus on flexibility and are more difficult to use. Because most manufacturing systems have some unique intricacy, the best packages allow the user to combine easy-to-use constructs with more flexible, lower level constructs. There are some packages that are particularly good at representing material handling or some other aspect of manufacturing processes. Simulation packages also differ in their support for both input and output data analysis.

The list of criteria for simulation package selection for emulation model building is long and most of them are important and commonly available in most packages for emulation modelling. Nonetheless, below are the criteria that are not commonly available but important to be considered for HSEM building project.

1) **Modular**. Modularity allows the user to develop the model in separate modules step by step. Each module can be tested and debugged separately and then linked together. The merging of models when a previously made model is going to be a sub-model for a larger model is useful. This option would be further enhanced if a library of reusable modules and pre-existing generic models is available.

2) **Coding aspects.** Possibility to enlarge the flexibility by adding user code to the simulation models either via external codes like DLLs or with user methods and event Access to the source code of the simulation software is useful when integration requires programming. A library of in-built functions and the possibility of defining functions by user further enhance this criterion.

3) **Integration with other application**. Simulation software may integrate with other packages such as spreadsheets, statistical packages, database management systems, CAD, and word processors to import or export data.

4) **Speed control.** Control of the speed of the model run is a desirable feature. One can see the flow of the model better at a low speed and use it for debugging, while he/she can save time by running the model in a high speed mode. For real time (RT) simulation facility which is essential for emulation, some simulation packages have standard built-in features for real time progress in simulation models, while some requires developing a customized synchronization tool.

5) **Open Architecture**. The simulation package should allow the modeler to model complicated control structures. It should be possible to implement complex logistic rules and control algorithms. Packages that offer interface to

programming language like VBA or C++ as well as OPC compliant are considered to have a clear advantage.

6) **Animation**. For emulation real time viewing is necessary. Whether true to scale or iconic or whether 2D or 3D the requirement depends on the objective of the project.

To illustrate the criteria for selecting simulation software for a HSEM project eight simulation packages were selected for comparison. Selection was based on case studies and results of the questionnaire survey conducted as explained in Chapter 3. They are Arena, Automod, EmPlant (*Simple++*), Extend, FlexSim, Promodel, Simul8 and Witness. Below is the general description for each package.

Arena

Arena a product of Systems Modelling Corporation part of Rockwell Software Automation is a flow oriented simulation language with the basis language SIMAN. Arena is a graphical modelling/animation system that is based on hierarchical modelling concepts. It allows user to create new modelling objects called modules, which are the building blocks of model creation. Models are created by drag and drop modules in a large window. These modules represent one or more statements of the SIMAN language.

Automod

Automod, software from AutoSimulations Inc part of Brooks Automation has general model building features, including the specification of processes, resources, loads, queues, and variables.
Automod is a simulation package that keeps a special focus on the space that object require. The animation capabilities include true to scale 3D graphics, rotation and tilting. A CAD like drawing utility is used to construct the model.

eM-Plant

eM-Plant is a rename of Simple++ from Aesop Corp which has become part of Technomatix Technologies. Simple++ is a fully object-oriented simulation system with an integrated graphical user interface. The user creates models by making a library of objects. These library objects represent classes (or parents) whose instances (or children) can be inserted into the models. Simple++ takes advantage of the features of object orientation, including class structure, inheritance, hierarchy, modularity, and polymorphism. In addition, Simple++ has an open architecture that allows it to communicate with other software.

Extend

Extend, from Imagine That Inc. is a visual, interactive simulation tool. Extend contains a built-in development system that allows the user to construct components and build custom user interfaces. Models are constructed graphically by dragging and dropping blocks (high level model components) from library windows onto the model worksheet. Other features include suite of inter process communication tools, hierarchical modelling capabilities and built-in optimization package.

FlexSim

FlexSim from Flexsim Software Products Inc is a Windows-based, fully object-oriented simulation environment for modelling discrete-event flow processes like manufacturing, material handling, and office workflow in 3D virtual reality animation. Models are created graphically, using drag and drop ready-made model

Promodel

Promodel from Promodel Corp is a manufacturing-oriented discrete event simulation software, used for evaluating, planning or designing manufacturing, warehousing, logistics and other operational and strategic situations. These products are Windows

based applications with intuitive graphical interfaces and object-oriented modelling constructs, eliminating the need for programming.

Simul8

SIMUL8 from SIMUL8 Corp is a general-purpose graphical oriented simulation package with a point-and-click user interface used in a wide variety of applications. Simul8 claims that the Windows based graphical interface allows user to build relatively complex models without needing to learn a programming language.

Witness

WITNESS is a Windows applications developed by Lanner Group has been used across a wide range of business applications both in the manufacturing and service industries. Key features of the WITNESS approach include building block design, modular and hierarchical structure, range of logic and control options, comprehensive statistical input and reports and openness to link the system to other software such as CAD, BPR, mapping tools and spreadsheets.

Several sources of information exist that provide good descriptions of simulation software from various aspects, including plain descriptions of popular packages and languages used in simulation; similarities and differences between packages ((Banks et al. 2000);(Klingstam and Gullander 1999)); what to consider when selecting a package (Nikoukaran and Paul 1999);(Tewoldeberhan, T. W., A. Verbraeck, E. Valentin, et al. (2002); user requirements surveys (Hlupic 2000); and updated lists of current version and price of the most popular packages.

The comparison of features against the simulation software packages selected is shown in Table 4.1. The comments, included where available, were gathered from conference papers, product brochures, website information, demo CDs, email correspondence and discussion with some software vendors and users.

Papers offering descriptions of some simulation languages and environments include the following: Arena (Bapat and Sturrock 2003), AutoMod (Rohrer and McGregor 2002), eM-Plant(Heinicke and Hickman 2000) , Extend(Krahl 2003) , Flexsim (Nordgren 2003), Promodel (Harrell and Price 2002), and Witness (Rawles 1998).

It has to be said that the list and comments are by no means complete and it does require regular review. Nonetheless it can be used as a guide and basis for further research particularly in choosing the right software and tools for HSEM project. Clarification on the comments presented and updates can be found on the sources and materials mentioned earlier and latest versions of product brochures and websites.

Some general comments to be noted from the software comparison are as follows.

1. Modular modelling is quite well supported in current simulation software packages.
2. Accessibility to codes varies among packages. While object oriented simulation packages offer truly openness, other packages offer limited access to codes in their own languages or C++.
3. While some packages provide built-in facilities for some specific function like real time modelling and interfacing, some packages require a development of customised functionality or module for assistance.
4. Although some capabilities are claimed to be featured in their respective simulation environments, the extent of readiness and ease to use is still subjective.

Table 4.1 Comparison of Features for HSEM against Selected Simulation Software.

	1	2	3	4	5	6	7	8
	Arena	**Automod**	**eM-Plant**	**Extend**	**FlexSim**	**Promodel**	**Simul8**	**Witness**
Modular	Yes, Object based	Yes, Object based for MHS components	Yes, Object oriented	Yes, Object based	Yes, Object oriented	Yes, Object based	Yes Object based	Yes Object based
Coding Aspect	Yes Access to SIMAN blocks	Yes in source file coded in Automod text editor	Yes	Yes	Yes written in C++	Yes, written in C++	Yes, in-built language *Visual Logic*	Yes Code editor Plain text
Integration with other application	Yes,	Yes MCM (Model Communication Module)	Yes	Yes	Yes	Yes	Yes, Visio	Yes FACTORYCAD
Speed Control	Yes , with Real time facility	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Open Architecture	Yes, VBA, C++ and OPC COM interface	Yes, VBA, C++ and OPC	Yes, VBA, C++ and OPC COM interface eM-Plant socket	Yes built-in compiled language (ModL)	Yes	Yes VBA	Yes VBA COM interface XML interface	Yes HLA compliant
Animation:	Iconic, 2D	True to scale 3D	3D	2D iconic	True to scale 3D	Scaled 2D	Iconic 2D	Iconic 2D

4.5 Summary

This chapter has discussed the technical framework and specific requirements for HSEM building including structural requirement, construction approach as well specific simulation software features for a HSEM project.

In general, flexibility of adding details to the simulation model while maintaining its correctness and facilitating communication with the control system underlines the requirement and approach towards developing such a model.

Flexibility requires the model to be adaptable to changes as well as structurally modular. Simulation model adaptability is the ease with which a simulation model can be modified, either to conform to changes in the system it represents, or to demonstrate the effect of changes to the system. Modular means that changes are local and are independent of the rest of the model, since they are encapsulated within a single module. Modularity supports the re-use of model components. It also provides a way of introducing hierarchical components into simulation models.

The general modelling approach that is recommended is to remain at the highest level possible when creating the models. However, as soon as that these high-level constructs do not allow capturing the necessary detail, drop down to the next level for some parts of the model rather than sacrifice the accuracy of the simulation model.

Inter process communication allows the integration of external data and applications into and out of the simulation models. In the context of emulation, communication between the model and the real controller has to be established and synchronized. Real time modelling capability and model communication interface are the basic requirements for a HSEM project.

Several features of simulation software those are essential for a HSEM modelling has been identified. Comparison among the selected software based on those features has indicated that, in general, HSEM is viable. Some software packages are more ready the others, other packages require some developments or modification and inclusion of appropriate functionalities or modules.

The development of HSEM as in other simulation projects does not depend on technical requirement alone. It requires the involvement of broad spectrum of expertise and activities. Chapter 5 describes the novel methodology or system approach that covers the technical requirements as well as organizational issues to develop HSEM.

CHAPTER 5

SIMULATION-EMULATION CONVERSION METHODOLOGY

5.1 Introduction

The previous chapter has described the technical requirements for developing Hybrid Simulation Emulation Model (HSEM) which include flexibility of adding detail , inter-process communication and specific emulation modelling features that a simulation software package need to have for such a project. This chapter describes, in two parts, the system approach to develop such a model.

The first part discusses the general methodology that can be used in a general application area. Section 5.2 outlines the prerequisites for using this methodology, Section 5.3 discuss the framework of such a project. Section 5.4 describes the phases and steps of the methodology.

The second part explains the specific elements for emulation model building. A manufacturing system development is used as example. Section 5.5 suggests an

approach towards modelling different level of detail. Section 5.6 demonstrates a method for interfacing between model and external controller.

5.2 Prerequisite for HSEM Methodology Deployment

As with any project, using a methodology alone does not solve all problems in an emulation project. Three important aspects to consider for a successful project are (i) the emulation project team should involve a broad spectrum of employees, from shop-floor operators to key decision makers; (ii) the emulation analysts must have good knowledge of simulation methodology and programming, and (iii) selection of the right simulation software tools.

A simulation project team usually comprise of (a) model project owner or client group consisting of managers and system users (for example AMHS operators and technical support) and (b) simulation developer group consists of project manager and modellers. However, due to the need to integrate the real control system with the model, emulation project team need to include process controllers, control engineers and software engineers.

5.3 Framework of Simulation-emulation project

The philosophy of the methodology centres on making emulation modelling simple and flexible, requiring minimal (if any) programming and using available built-in technology. The software comparison summarised in Table 4.1 can be used as indication of the availability of relevant tools in the simulation software in the market today.

The proposed methodology is a synthesis and extension of many ideas and developments published , among others, covering the topics of manufacturing system

development (Wu 1994), simulation methodology (Eldabi and Paul 1997; Banks 1998; Sadowski and Grabau 1999; Law and Kelton 2000), real time simulation methodology (Lee and Fishwick 1999; Brennan 2000; Versteegt and Verbraeck 2002) as well as verification and validation of simulation model (Balci 2003).

The HSEM methodology is defined by a cycle, comprising the following activities:

a. *Model definition:* the user defines a model of the system to develop, splitting them in different subcomponents.

b. *Simulation and validation against the real system*: simulations are derived, and detailed behaviour is analyzed.

c. *Detailed experimentation in a virtual environment*: in order to ensure validity, different experimental conditions are tested.

d. *Development of the actual subsystem in a hardware surrogate*: simulated versions of the model are replaced by real-time executable versions. This is done by automatically replacing the simulator by a real-time engine.

This cycle is incrementally repeated providing feedback to change the models originally defined. The process stops when the system is fully developed and tested, and every simulated component has been replaced by an executable one. The process might result in modifying the models originally defined according to the simulation or execution results obtained.

5.4 Simulation-emulation model building phases

The HSEM methodology cycle defined in the previous section can be demonstrated by the flow of activities in phases shown in Figure 5.1. Even though some phases and steps are similar to the development of a simulation model there are important ones that are specifically essential for emulation model building. Those phases and steps

are highlighted with **bold** letters to illustrate the novel elements in the approach. Specific explanations are included to emphasise certain points in some steps in a simulation-emulation project.

5.4.1 PROBLEM DEFINITION (HSEM Phase 1)

The first phase of the simulation process has the most effect on the total simulation study since a wrong problem definition can waste a lot of time and money on the project. It is important that the problem definition should be explicit and documented as part of the Project Functional Specifications. This phase includes the following activities:

- Define the objectives of the study.
- List the specific issues to be addressed and the performance measures for evaluating a system design.
- Determine the boundary or domain of the study.
- Determine the level of detail or proper abstraction level.

The task of determining which components of the real system to include and exclude from the simulation model requires both insight on how the real system operates and experience in simulation modelling. A good method is to make a list of all components in the real system and identify those components that may have a significant direct or indirect effect on the simulation model output. For example, for the AGV material handling study one may include the following components of the real system: stations to be visited by the AGVs in order to pick up or drop parts, AGV path, and AGV battery recharge stations.

The information gathered at the end of this phase should suffice to estimate the total cost of the project. The simulation Group and the client generally meet / visit to observe the actual or a similar process during this phase is recommended too. A formal proposal is generally written at the end of this phase of the project. Continuing with the other phases is contingent on its acceptance.

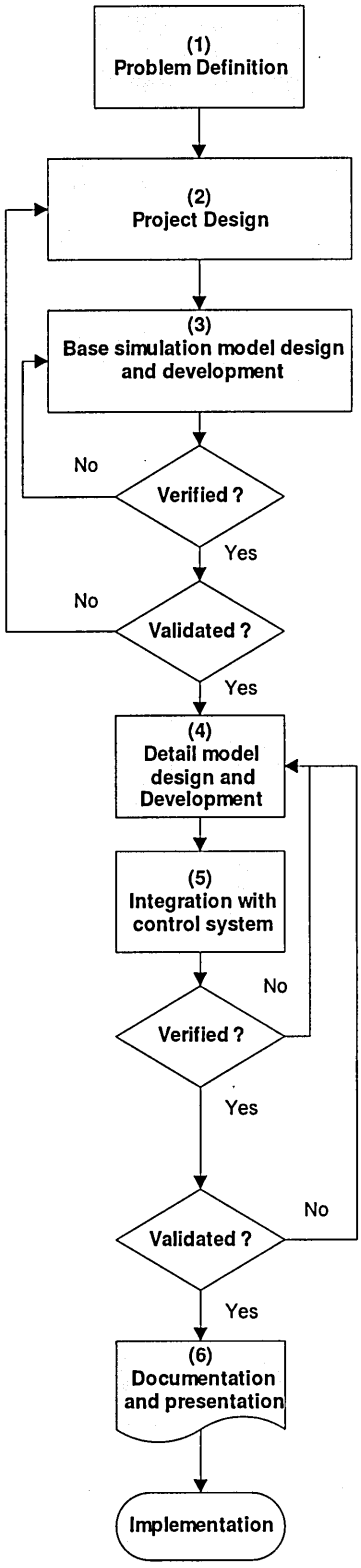


Figure 5.1 Overall Hybrid Simulation-emulation model building flowchart

5.4.2 PROJECT DESIGN (HSEM Phase 2)

In the second phase, some steps of the previous phase are investigated in more detail and the technical aspects of the problem are given more weight. The project team discusses issues in detail with the line engineers and operators. This phase includes the following activities.

- Estimate the life cycle of the model.
- List broad assumptions.
- Determine the animation requirements.
- Determine the level of data available and what data is needed.
- Determine the human requirements and skill levels.
- Determine the audience (usually more than one level of management).
- Identify the deliverables.
- **Check for simulation-emulation model viability conversion. Simulation-emulation conversion checklist is as in Table 5.1.**
- **Select simulation-emulation software package.**

Table 5.1 Simulation-emulation modelling software features checklist

	Features	Availability
1	Flexible modelling with variable levels of details (Modular, hierarchical and configurable)	
2	Accessibility to source code and adding user code to the simulation model	
3	Integration with other applications (Database, Statistical, optimizing tool, 3D graphics etc.)	
4	Real time modelling facilities	
5	Inter process communication capability (ability to cooperate and communicate with other software packages and real systems)	

The information gathered at the end of this phase of the project is documented in the Project Functional Specifications. The project team may change the project timing and resource requirements based on the new information available at the end of this phase of the project.

5.4.3 BASE SIMULATION MODEL DESIGN (HSEM Phase 3)

The overall strategy should focus on finding a model that minimizes the simulation effort while ensuring that all objectives of the project are met and all specific issues are investigated. The third phase includes all or part of the following activities.

- Determine the elements that drive the system.
- Determine the entities that should represent the system elements.
- Determine the level of detail needed to describe the system components.
- Determine the graphics requirements of the model.
- Build the basic simulation model.
- Validate the basic model.

5.4.4 DETAIL MODEL DESIGN AND DEVELOPMENT (HSEM Phase 4)

At the fourth phase, the modeller describes in detail the operating logic of the system and performs data collection and analysis tasks. This phase includes the following activities.

- Obtain the operation specifications from "subject-matter experts" (SMEs).
- Obtain the material handling specifications.
- List all information and data summaries in an "assumptions document," which becomes the major documentation for the model.
- **Identify the areas that utilize special control logic and build the control logic accordingly.**
- **Define the interfaces between the components.**

- Describe the process in detail.
- Use **sensitivity analyses** (Law and Kelton 2000) if necessary to determine important model factors, which have to be modelled carefully.
- **Modify the base model to the desired level of detail for emulation**
- **Modify the control logic accordingly.**
- Verify the detail model.

An important point to note at this phase is that for emulation modelling, the system to be analysed has to be divided into a (simulation) model of the plant (e.g. machinery) and a (simulation) model of the control system (e.g. PLC, PAC, DCS) in order to replace one component by reality. Therefore it is essential for the communication between control system and the plant is explicitly modelled at the earliest stage. This would enable the code to be developed and refined as the model is developed.

The information that is generated at this phase of the project may be used to create the Maintenance Manual, if one is requested by the client. In any case, this information can be integrated into a detailed Project Functional Specifications as well as into the Project Book and the model code.

5.4.5 INTEGRATION WITH CONTROL SYSTEM (HSEM Phase 5)

The integration of the emulation model with controller, its verification, and operational validation constitute the fifth phase of the process. This phase, shown in figure 5.2., includes the following guidelines and activities.

1. **Consider the tool availability and limitations for emulation, with reference to Figure 5.1.**
2. **Check control system structure and variables**
3. **Define the interface required.**
4. **Use built-in or existing interface as much as possible.**
5. **Build new interface(s) if necessary.**
6. **Integrate emulation model with control system**
7. **Verify communication.**
8. **Test run to validate emulation model**

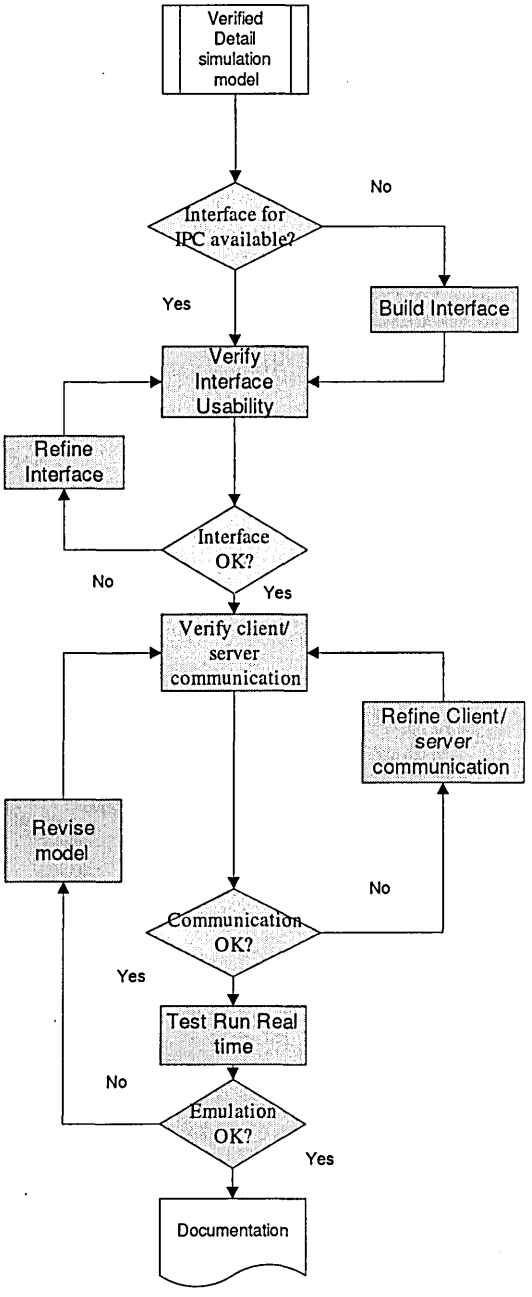


Figure 5.2 Emulation Model-Controller Integration Flowchart

This phase is specific for connecting emulation model to the real controller. The guidelines and steps are further explained in 5.5. The model obtained at this phase is ready for experimentation or test run in real environment.

5.4.6 DOCUMENTATION AND PRESENTATION (HSEM Phase 6)

Good documentation and presentation play a key role in the success of a simulation study. The sixth phase also applies to long-term emulation life-cycle studies where the models are maintained throughout the life of the real system. On the other hand, short-term simulation studies are those where once the simulation results are used in the decision-making, the model is not used any more by the client.

The long-term emulation studies require a long-time ownership of the model by a modeller and/or engineers that are going to use the model. One may categorize long-term life-cycle models into four categories in terms of use, namely; (a) training, (b) scheduling, (c) system redesign, and (d) launching phase analysis. Training models are built to train client personnel in emulation as part of a simulation class or to familiarize the new personnel in the system. Scheduling models are models such that when the product-mix and batch sizes change, the scheduling rules are tested under the new conditions for best resource utilization and product deliveries. These two categories may require no or minimum modeller follow-up once the model has been transferred to the client.

The models for system redesign and launch phase studies may require a close modeller-and-client-engineer interaction so that the model is not misused. System redesign models are used whenever a change in design of the system is to be implemented. Models that are used for launching phase analysis are those used during system launch to allocate resources (e.g., workers) effectively in the partial operation of the whole system. In many cases, long-term life-cycle models are used for multiple purposes including all four categories.

It is important that the long-term of the model usage should be identified as part of the original objectives of the study because the model design is highly influenced by it. A representative of the long-term users of the model should become a member of the project team right from the beginning of the study.

5.5 MODELLING DETAIL VARIABILITY

This section focuses on the process of modelling the different levels of detail or *multi-resolution modelling* of the Manufacturing System Design (MSD) and how such models may be built in a manner that helps the already existing simulation packages to build a base simulation model and then flexibly extend it to the detailed design model for emulation.

Figure 5.3 illustrates the modelling concept of integration of different levels of detail. An example would be the same manufacturing system can be modelled in different ways in three different simulation models. The models can be built with different levels of detail. The first model may contain all machines, servers, conveyors, and buffers present in the system. This model is considered to be having high level of detail. The second model can be a lower level of detail, but keeps the notion of manufacturing lines without the possibility to store products in between servers. The third model can be built with a very low level of detail. Here, complete production lines cab be built as single servers, producing complete batches.

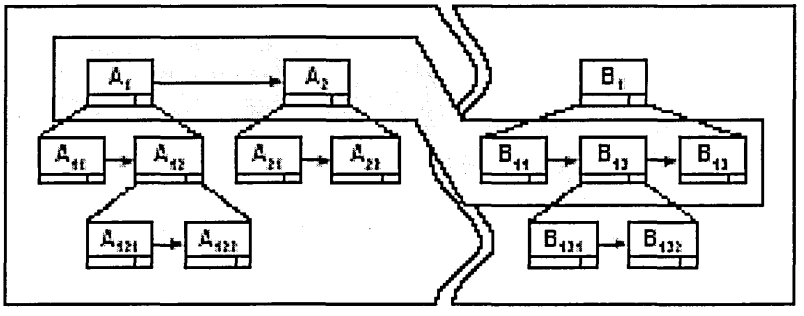


Figure 5.3 Appropriate Level of Detail for Integration
(adapted from (Benjamin et al. 1998)

A simple method is introduced here which is based on some guidelines to ease the process of modelling systems with variable details. The following subsections describe the main steps for this method.

Modelling the Conceptual Level

1. Identification and Classification of Model's Entities:

Considering the conceptual level of the MSD, the first major step in building such a model should be the identification of the building blocks of the system, such as the different types of cells. When starting to identify the basic components of the conceptual model, the modeller must bear in mind that this model is to be extended into more detail later in a flexible manner without the need to create a new detailed simulation model from scratch. Therefore, the major components of a conceptual model might be classified as separate, preferably non-overlapping, blocks or entities regardless of their internal structures and details.

2. ***Assigning Entities Activities***:

After the identification of the main entities of the model, the second step is to assign the behaviour of each entity.

Generally, the modeller, when developing the conceptual model, must avoid including any unnecessary details that may overcomplicate the conceptual model. On the other hand, forgoing any other important components at this level will increase the problem of complication in the more detailed stage. If a simulation model of the conceptual level is built correctly, it will provide the required results and at the same time it will be a well established base for detailed design. This can easily be extended with more details and complexity. At this stage the modeller may assign equal numbers for each entity or resource. For instance, he/she may assign the same number of machines for each cell. Another example might be the assigning of equal speed of transportation between any two cells. This is to eliminate the effect of such details on the simulation results. Generally speaking, the model at this stage is not necessarily 'valid' or typical of the real system.

Modelling the Detailed Level

3. *Entering Model's Details*:

At the detailed design level, the third step is to enter the new details within the boundaries of the blocks, which are already created at the conceptual level. That is, each block of the conceptual model is expanded separately from the rest of the model. Detailed data of a cell block could be the number of machines in the cell, the process duration of each machine, rate of failure for each machine, and maintenance time. Sometimes it can be expanded into an internal network of activities. For example, in a 'painting-cell' block parts may be queued for cleaning, then after cleaning they are transferred to another queue for painting. Some details might be entered as interactions between different entities such as, physical positioning, distances, and directions between cells within the system.

4. *Re-Assigning Entities' Details*:

At this level, information assigned at the conceptual level is to be reassigned by introducing the real values to each entity before fine-tuning it to achieve the best results.

The detailed design level can be considered as a network of blocks, each block containing all its corresponding details and other necessary details which represent interactions with other blocks. In addition, it gives the real physical layout. It is worth noting at this stage the model validity is very important, that it should represent all the details that make up the system as accurately as possible.

Generally, this method of classification will ease the process of model building and reduce the chance of error, as all necessary modifications are to be made from within the entity's boundaries with no subsequent effects on other parts of the system. This reduces the time needed for any changes to the model, as a change of one entity will not affect the rest of the model. This method can be considered useful for effectively building flexible models with variable levels of detail.

The above discussion gives an overview of how system components can be classified for flexible modelling of detail variability at the conceptual level, then how data is arranged and reassigned at the detailed level to be entered into the simulation package.

5.6 MODEL COMMUNICATION WITH EXTERNAL CONTROLLER

Results from the survey conducted among the simulation practitioners in the earlier part of this research has indicated that interfacing between models to be the most required facility to ease emulation model building (Hasnan et al. 2005).

A 'plug and play' interface, without writing any program code, is a desired feature. There are tools, the like of RT-Lab LabVIEW API tools and Automod Model Communication Module (MCM), in the market that provide assistance for developing interfaces between proprietary applications and devices, but the usage is rather limited, mostly due to lack of standard specifications. Data Access Specification developed by the OPC Foundation is seen to have the potential to make such feature. Until that happens, an emulation modeller needs to do some form of programming to build a suitable interface.

This section outlines the steps to establish the communication between emulation model, in this application regarded as 'client' and the external controller which is regarded as 'server'.

There are a number of transport protocols to move packets of data from client to server that a modeller can choose, normally based on the platform being used. Among them are Novell's IPX/SPX, Apple AppleTalk, Transmission Control Protocol/ Internet Protocol (TCP/IP) and Open System Interconnection (OSI). There are also various client-server protocols to dictate the manner in which clients request information and service from a server and also how the server replies to that request. Examples are NetBIOS, Remote Procedure Call (RPC), Advanced Program-to-

Program Communication (APPC), Named Pipes, Sockets, Transport Level Interface (TTI), Sequence Packet Exchange (SPX) and OPC.

Studies on features simulation software, as discussed in Chapter 4, has shown that all simulation packages provides mechanisms for external program to interact with the simulation through VBA module and Dynamic Link Library (DLL) usually implemented in C++.

The steps outlined below uses TCP/IP protocol and socket technology for MS Windows Socket called Winsock. Winsock is a set of routines that reside in a DLL interfaced with TCP/IP and from there through to the internet.

1. Create client application in the simulation model using a program editor. Usually the program editor is built-in or attached to the simulation software.
2. Create server application for the controller program.
3. Create message handler application or communication module to manage the transfer of messages.
4. Launch both client and server applications.
5. Verify connection.
6. Confirm the availability of real time modelling features in the software package. If not, create application tool to enable real time run.
7. Run the emulation model in real time.

The programming details to create the client and server applications are not within the present scope of the thesis. Nonetheless books and websites that teach and discuss the programming side are available and too many to list. However two books that have been helpful in this research are (Horton 1998) for programming in C++ and (Wright 1998) for programming in Visual Basic.

An important issue concerning the communication between emulation model and external controllers is the synchronization. As (Versteegt and Verbraeck 2002) reported that whilst certain software (e.g. Arena and Simple ++) offer standard built in feature for real time progress, other software (e.g. Automod) requires the construction of a customized tool, in their case called 'wall-clock peeker' to

synchronize the simulation clock with the wall lock every fixed time unit. It could be implemented in a user written C++ function in a DLL.

The general procedure is also applicable for using OPC as the means for communicating between the emulation model and the controller system in a multi client/server situation. It can also be used as a basis for application in a distributed simulation environment.

5.7 Summary

This chapter has described the system approach to develop the Hybrid Simulation Emulation Model (HSEM). The summary of general steps for HSEM modelling is shown in Table 5.2

While emphasizing on the technical issues this chapter has also covered briefly the non technical aspects of the development including project planning and some organizational issues. As for documentation, although it is essential for every phase of the project, is not specifically discussed in this chapter as the process is taken to be the same with any simulation project.

Some highlights in this chapter include:

1. A new methodology is proposed to accelerate emulation model building through an efficient hybrid approach. This approach should minimise the effort required to build emulation models.

2. The HSEM project requires a team comprising of multidisciplinary expertise. The major difference from the usual simulation project is the inclusion of expertise in the field of information and communication technology (ICT) and control engineers, particularly to develop the interface between the emulation model and external controllers.

3. For emulation model, synchronization is essential. Since the objective of HSEM project is to develop emulation modelling simple, a built-in feature for real time modelling is required.

4. The methodological approach of multi-resolution modelling detail would be useful as a guide for the model builder to develop the emulation model efficiently.

Table 5.2 Summary of Steps of HSEM Modelling

Steps	Summary Procedures
Base Model	
Step 1	Identification and classification of the main blocks or entities of the system separately to be extended into more detail later.
Step 2	Assigning averages and assumptions of real data to the established blocks and not entering much detail.
Detailed Model	
Step 3	Adding more extensive details (entities and activities) needed to build final model including all necessary factors such as physical layout.
Step 4	Reassigning the model's behaviour by entering the real data into those blocks then fine-tuning the model to achieve the required results.
Control System Integration	
Step 5	Define interface required, build new interface if necessary and verify usability.
Step 6	Integrate emulation model with control system and verify communication.

The validation of the methodology presented in this chapter is discussed in the following chapter. Chapter 6 describes the development of HSEM for a small

manufacturing, progressing from a simple basic model to more elaborate model involving external intervention to the emulation model.

CHAPTER 6

VALIDATION

6.1 Introduction

The previous chapter has described a methodology to develop a new type of model called Hybrid Simulation Emulation Model (HSEM).

This chapter reports the validation process of the proposed methodology. It describes the developmental work of using the methodology to build a hypothetical simulation model of a small manufacturing plant and converting it into an emulation model where some form of external control is included. Since the focus is on the emulation building, some parts of the work that are common with simulation model building are not discussed in detail. The description of the model building is simplified so as to highlight the conversion simulation-emulation aspects of HSEM building rather the output of the model. A human machine interface is developed and used to describe the interaction between emulation model and human controller.

The validation work begins, as reported in section 6.2, with describing the background of the plant and the aim of the modelling work. Section 6.3 highlights some important features of Arena Modelling for HSEM model building. Section 6.4 describes the development of the base simulation model and modelling the details of certain components and process in the model. Section 6.5 describes the integration of external controller with the emulation model. Section 6.6 reports the procedure HSEM was executed. Section 6.7 gives the summary of the validation process.

6.2 Project background and design

(HSEM Phase 1)

The management of a manufacturing plant aims to improve the efficiency and productivity on one of the production line for one of its products, without any additional capital investment.

A system developer team was assigned to develop a system, based on existing setup and resources that would improve the efficiency and productivity of the production line by 20%.

(HSEM Phase 2)

Results from preliminary study conducted by the team indicated that a major source of ineffectiveness came from large amount of non moving Work-in-Process (WIP). The system developer team then decided that a simulation study is to be conducted with the following objectives.

1. To identify problem areas in the production flow line.
2. To assess the current performance with regards to WIP status at each station.
3. To develop possible methods to reduce WIP.
4. To evaluate the effectiveness of the optimization strategies.
5. To present potential methods and improvements to the management.

Since some strategies that would be tried on the system involves integration of components with multiple level detail as well as some form of control algorithm, and also the model is intended to be reuse, HSEM is seen to be a viable option.

Further checks on the suitability of simulation software, as shown in checklist in Table 6.1, the team found that the Arena 7.0 was equipped for such project.

This simulation language was selected among others due to its ability to operate in conjunction with a real time (RT) package. Moreover, Visual Basic for Applications (VBA) is an integral part of Arena 7. This enables convenient access to databases and the automating of Arena models.

Table 6.1 Simulation-emulation modelling software features checklist for Arena

	Features	Availability
1	Flexible modelling with variable levels of details (Modular, hierarchical and configurable)	√
2	Accessibility to source code and adding user code to the simulation model	√
3	Integration with other applications (Database, Statistical, optimizing tool, 3D graphics etc.)	√
4	Real time modelling facilities	√
5	Inter process communication capability (ability to cooperate and communicate with other software packages and real systems)	√

6.3 Features of Arena Modelling

Below are some important points with regard to modelling in Arena SE version 7.0 that would be helpful towards understanding the HSEM model building explained in the sections that followed.

- SIMAN is a general-purpose simulation language that builds upon early languages such as SLAM
- In Arena the structures of SIMAN are implemented as graphical modeling objects
- Simulations models are built by placing these objects on a drawing board and linking the objects to define the model logic (basically a flow-chart of the system is constructed)
- In simulations the Arena model is parsed into SIMAN code, then compiled and executed
- The modeling constructs in Arena are called *modules*
- Arena contains a wide variety of modules that are organized into different libraries called *templates* or *panels*
- All modules are composed of SIMAN components, different amounts of functions are aggregated in higher and lower level modules
- Modules of different hierarcy levels can be used interchangeably
- New templates and modules can be created by using existing components or user-written VB, C/C++ or FORTRAN code

Figure 6.1 below shows the different hierarchy levels of the modules in Arena. (Kelton et al. 2004)

Data exchange between an ARENA model and the external data source are achieved by Visual Basic for Applications (VBA). The external data source can be database, spreadsheet and files in various formats. Procedures and functions can be defined through writing codes in the Visual Basic Editor corresponding to each VBA block in the simulation model logic view. When an entity arrives at the VBA block, procedures and functions defined in this block are fully executed. After everything is done, the entity may leave the VBA block and go on to the next module/block.

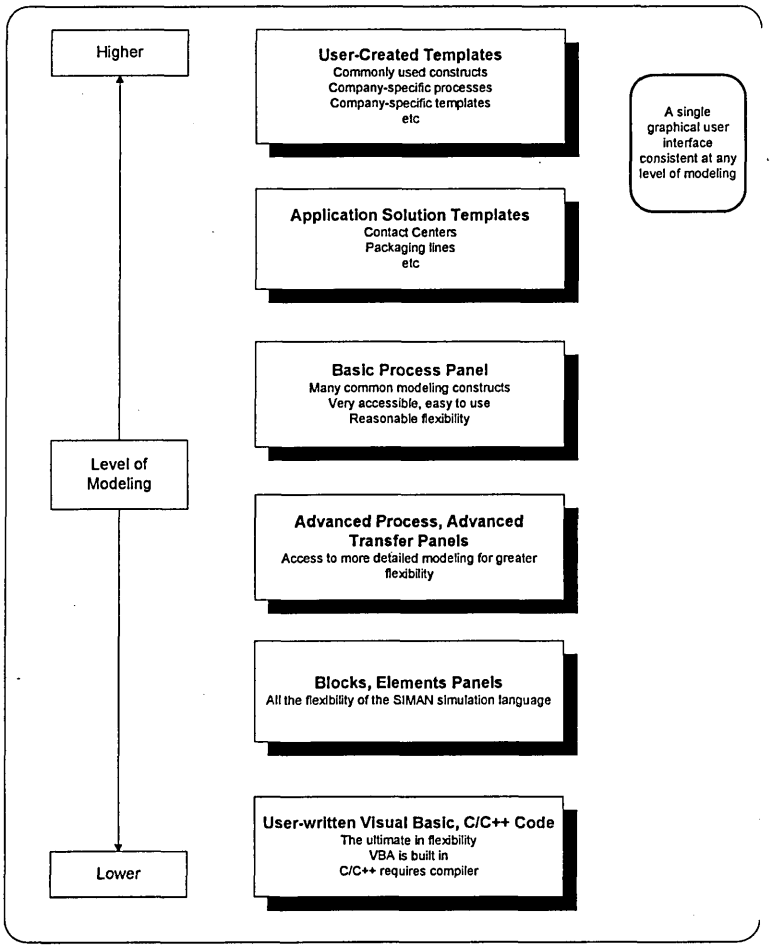


Figure 6.1 Arena hierarchical structure

Another feature of Arena, called Arena RT can be used for a simulation model to interact with external client applications. This interaction is performed via an online messaging system. For example, the simulation model might contain aggregate-level system logic that sends tasks in real-time to a facility's shop floor control system. In this case, Arena's client might be a messaging queue that interfaces directly with PLCs. After completion of this operation (automated or manual), a message is sent back to the model so that the simulation can be updated and further instructions can be issued. During the execution of the model, the simulation and actual shop floor could operate concurrently. The animation could serve as a real-time monitoring device.

6.4 Model Building of a Manufacturing Plant

The production side of the manufacturing plant is made up two staging area and three processing cells. The two staging areas are divided into part arrival station and product exit station. The three processing cells, arranged in series, are the machining, painting and packaging centres. The layout of the plant is as shown in Figure 6.2.

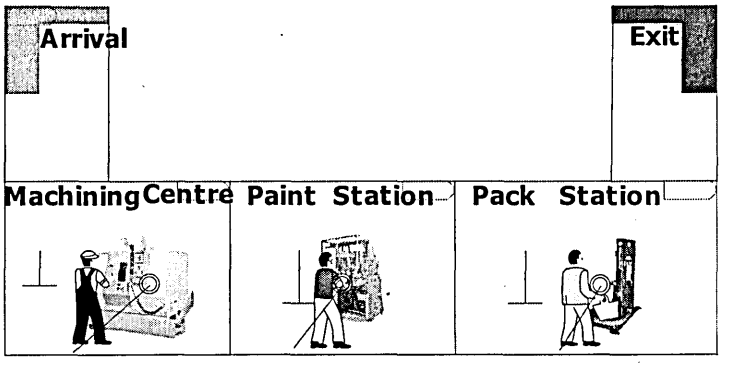


Figure 6.2 A Manufacturing plant layout

Parts entering the system are placed at a staging area of the Arrival Station, for transfer to the first workstation, a machining centre. After the parts have completed processing at the machining centre, they are transferred to a paint station manned by a second worker, named painter, then to a packaging station where they are packed by a third worker, named packer, and then to a second staging area, Exit Station, where they exit the system. The transfer of parts between stations is by means of an Automated Guided Vehicle system (AGVs).

6.4.1 Base Simulation Model

(HSEM Phase 3)

The basic or conceptual model consists of one machine per cell with each machine having different processing time. The purpose of the model is to estimate the following performance measure.

- Throughput
- Time in system for parts
- Times parts spend in queues
- Queue sizes
- Utilization of equipment

Since the transfer behaviour was not required, the ROUTE module, a type of Transfer block, was used to model the unconstrained movement of entities from one station to another. The ROUTE module or connect method used assumes that time may be required to move the entity between stations, but it operates on the assumption that no additional delay will be incurred because of unavailable resources or transporters.

The model logic, in the form of Arena flowchart type modules which also define the routing of simulation entities through the system is shown in Figure 6.3.

The modelling uses modules in the Basic Process Panels and Advanced Process, Advanced Transfer Panels, viewed as middle level in the Arena hierarchical structure shown in Figure 6.1.

As shown in Figure 6.3, the model logic can be summarised as follows:

1. Create entity called `part` at the `Arrival Station` at random `Time between arrival` type specified in the CREATE module.
2. Assign `Time In` variable for part entry time into the system.
3. Transfer part from `Arrival Station` to `Machining Centre`. ROUTE was used to allow modelling transfer of entities between stations, with a defined time delay in the transfer.
4. Part arriving at the machining centre is put in Queue before being processed.
5. In the PROCESS module, the entity SEIZE the resource, in this instance the `Machinist`, DELAY for specified processing time and RELEASE the resource.
6. Part is transfered using ROUTE to the `Paint Station`.

7. Steps 4 to 6 are repeated for processing of entities at `Paint Station` and `Pack Station` respectively.
8. Part is transfered using ROUTE from the `Pack Station` to `Exit Station` where entity exit the system. The time interval during which the entity remains in the system is recorded as `Flow Time`.

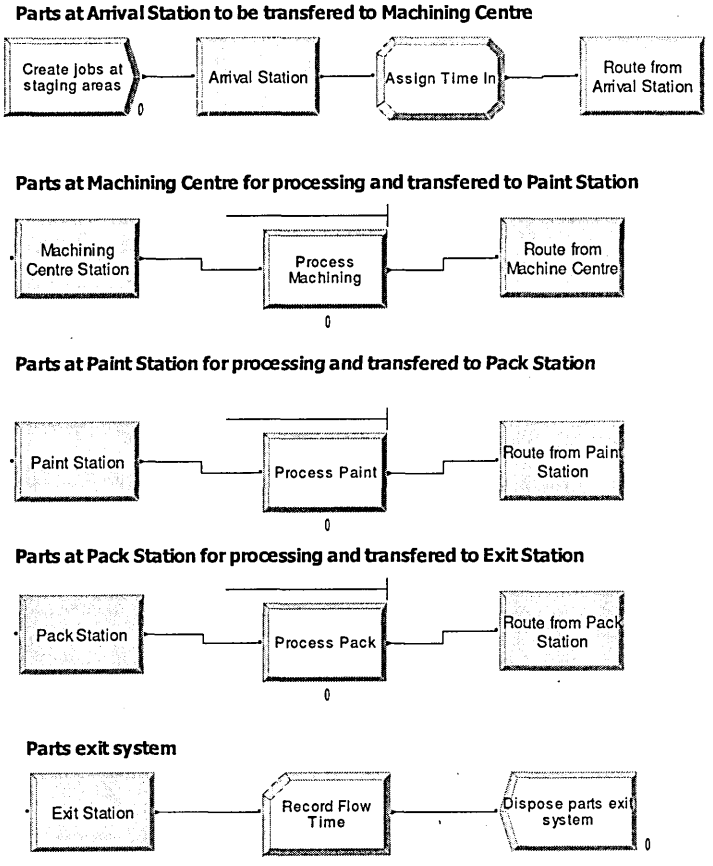


Figure 6.3 Model Logic for Base Simulation Model

The animation level was set to sufficiently indicate flow of entities between stations. The animation is as in Figure 6.5.

6.4.2 Modelling Details

(Phase 4 of the methodology)

After a review it was found that to improve performance, some changes and detailing need to be made.

At the next phase of study, the management and system developer have identified two areas in the system that can be re-develop with minimal cost that could reduce the overall WIP. Those were (a) loading and unloading of parts between every station and the movers, and (b) transfer of parts between stations.

A more detail study on the effect of any changes made in the two areas on the whole system has to be made. The base simulation model developed earlier was then upgraded by adding *resource-constrained* modules in the form of entity transfer as well as loading and unloading process modules.

There are variety of approaches that can be used to add details to the simulation model, depending on factors such as involvement of resources and controllers, hierarchical level of modules etc.

Modelling the details of loading and unloading process can be done using PROCESS module which would consider the entity seizing a resource (loading or unloading operator or machine), delaying for loading or unloading time and release the resource. If the activity of the resource is not considered significant, using a DELAY module or block would be sufficient. In the present context, to keep the model simple yet flexible, DELAY module is used.

Modelling transfer of parts or material handling can be divided into two categories (1) based on the number of individual material handling device available, (2) based on space availability.

The transfer of parts between stations using the AGV, from a modelling standpoint, falls under the first category of moveable resources, referred to in Arena as transporters.

Arena provides two types of modelling transporters: Free-path and Guided. Free-path transporters can move freely through the system without encountering delays due to congestion. Guided transporters are restricted to moving within a predefined network. The travel times depend on the vehicle's speeds, the network paths they follow, and potential congestion along those paths.

The transfer of a part with a transporter requires three activities: `*Request*` a transporter, `*Transport*` the part, and `*Free*` the transporter. The `*Request*` activity, which is analogous to seizing a resource, allocates an available transporter to the requesting entity and moves the allocated transporter to the location of the entity, if it's already not there. The `*Transport*` activity causes the transporter to move the entity to the destination station. The `*Free*` activity frees the transporter for the next request, much like releasing a resource. The modeller can choose to use modules in Arena Advanced Transfer panel or SIMAN codes in the Blocks, Elements panels, depending on the approach and level of detail required for the model. The present model uses Free-Path transporter type with REQUEST, TRANSPORT, STATION and FREE modules from the Advanced Transfer panel. Other alternatives would be using LEAVE and ENTER Arena modules or REQUEST, ALLOCATE, MOVE etc from SIMAN blocks, element panels.

The model logic employed for detail simulation model of the manufacturing plant is shown in Figure 6.4.

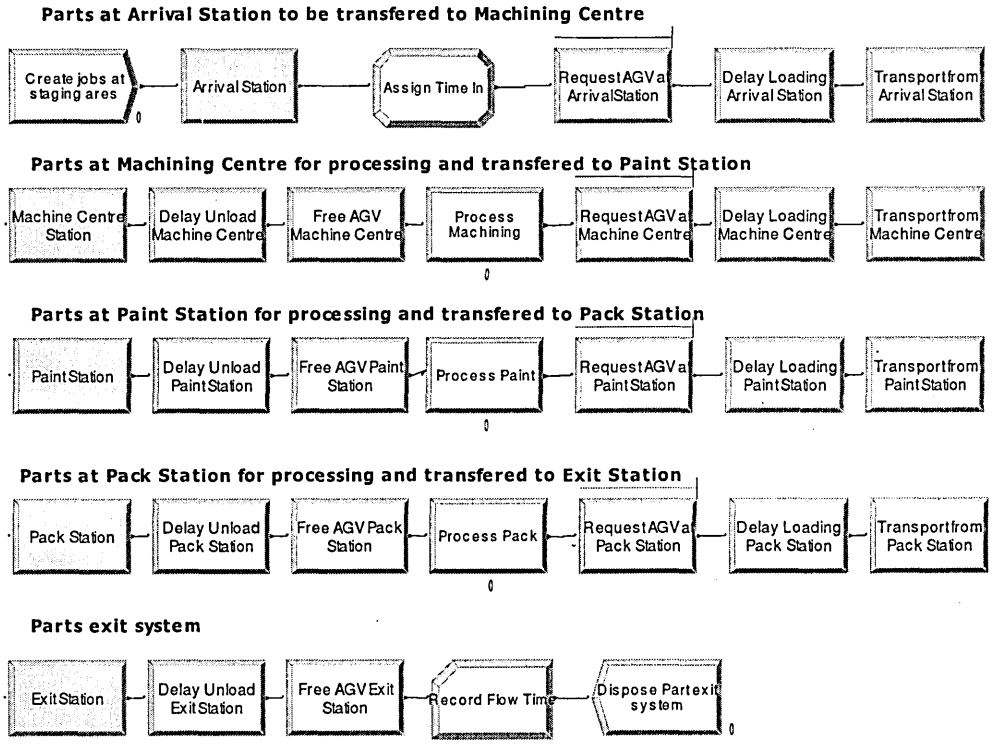


Figure 6.4 Model Logic for Detail Simulation Model

The model logic for activities in the transfer of part between stations can be summarised as follows.

1. Part request for an AGV.
2. If available, an AGV is allocated and move towards the part.
3. Part is loaded on to the AGV in a specified delay time.
4. Part is transported at specified speed to the next station.
5. At the next station, part is unloaded on to the next resource in specified time delay.
6. Part frees the AGV.
7. AGV waits for next instruction.

In the case of multiple AGV in the system, two situations regarding assignment to parts are possible. First, a situation during the run where an entity requests an AGV and more than one is available. In this case, Arena provides the choice of Transporter Selection Rules to determine which one of the transporter units will fulfil the request.

The second is when a transporter is freed and there are multiple entities waiting. In this case, Arena applies a built-in priority rule. The situations described highlight the use of built-in control logic options in the modelling of entity transfer. It is also to be noted that it is possible to separate the process and control logic using ARENA's simulation language SIMAN.

The animation for the detail simulation, shown in Figure 6.5, has included the graphic representation of the AGV movement.

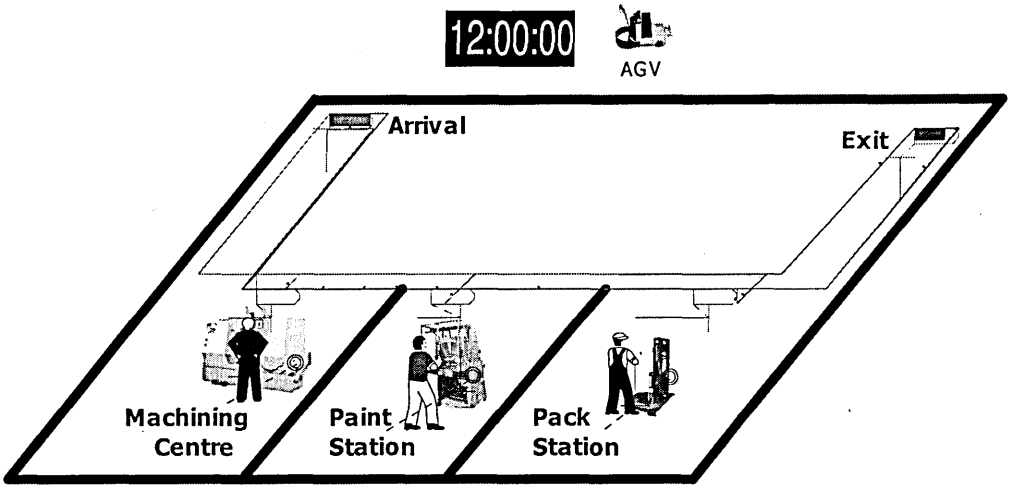


Figure 6.5 Animation for Detail Simulation

It is acknowledged that various issues can be studied and corresponding components can be detailed accordingly. Those include parts transfer logistics, scheduling strategies, track layout, parts order control, loading/unloading mechanisms and operating procedures. The areas in the system to be studied mentioned earlier can be modelled to more detail level using SIMAN codes, albeit requiring a considerable level of modelling skill. For the purpose of illustrating the process of conversion from simulation to emulation model building, the current detail simulation model is sufficient to be regarded as an emulation model. The next section discusses the steps to integrate the emulation model with external controller as a demonstration of completing the process HSEM building.

6.5 Integration with Controller

(HSEM Phase 5)

So far, the modelling logic and animation of the manufacturing system are based on the assumed time projection of events. In reality assumed time projections are not enough, unexpected events may occur. Therefore real time communication is sometime necessary. For example, the modeller may find in the detail modelling process, due to limited buffer space the machines need to communicate between each other as well as with the arrival controller to monitor the number of parts coming to the respective stations.

The next stage of the modelling needs to coordinate the simulation logic with the external process of a real system. In this model, the external processes considered as 'server' and Arena model considered as 'client' communicate via a messaging system, whereby entities in the Arena model send messages to the external applications to indicate simulated tasks, and the external applications send "message responses" back to Arena to indicate the tasks have been completed.

6.5.1 Emulation Communication Structure

The client/server communication between machine and computer in this model uses socket, a program device, which supplies sending, and receiving data via the defined TCP/IP port. In this case, socket technology for MS Windows called Winsock was used.

For ease of maintenance and flexibility, the non simulation model components are all implemented using the common programming framework of Microsoft Visual Basic for Applications (VBA). The implementation of the major supporting components within VBA not only allows for a powerful implementation using a variety of Microsoft products, but it also allows for direct links to those objects that are

exposed within Systems Modeling Corporation's Arena simulation software, version 7.0. The Visual Basic object library within Arena was used to incorporate significant flexibility into the model.

A key advantage to utilizing the VBA architecture within HSEM is its widespread availability in industry applications, not to mention the ease of use of its development environment. Readily available, off-the-shelf software products were used to provide the base functionality required within the modules of HSEM without reinventing or duplicating the significant development momentum provided by Microsoft and other software vendors.

A message handler application called RTConsole.exe was written in Visual Basic, acts as interface between client and server computer.

The emulation communication structure is shown in Figure 6.6.

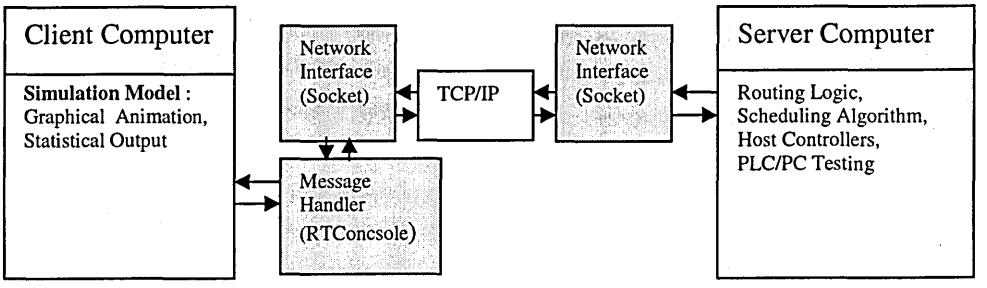


Figure 6.6 Emulation Model Communication Structure

ARENA RT which supports real-time synchronization and message exchange over TCP/IP was used to synchronize simulation time with real-time and to access the external controller. ARENA RT also supports switching between internal simulation logic and external control logic simplifying the process of separating classical simulation logic and simulation logic based on external control. Linking into the event scheduler of the simulator by one of the provided functions it was possible to

synchronize the simulation clock as well as to update the internal simulation data with the external I/O signals.

6.5.2 Implementing Changes

The following are the three major changes involving Arena and external code that are required to implement the inter-process communication (IPC) between the model and the controller.

6.5.2.1 Modification on Arena Object

To coordinate simulation logic with the external process of a real system, the simulation model needs to be programmed for the control system to open socket connections and read and send messages. This is done in VBA using the VBA events provided in Arena's ThisDocument object. The ThisDocument object gives the VBA project access to various events within the Arena model. To add code for an event procedure, one can select the *ModelLogic* object in the Visual Basic Editor, and choose the desired event (for example RunBegin) in the procedure list.

The following VBA events provided in Arena's ThisDocument object are used. These events are only called when Arena is running in execution mode.

- ***RealTimeInitialize***—called at the beginning of the first replication. Place code that initializes the inter-process communications here. It is also used to display userform (called frmConnect) to prompt the IP address and port.
- ***RealTimeSend***—Called when an entity tries to send a message to the external process. Code that sends the message to the IPC queue is placed here.
- ***RealTimeReceive***—Code that receives messages from the IPC queue and passes them to Arena.
- ***RealTimeTerminate***—called at the end of the last replication. Code that terminates the inter-process communications is placed here.

UserForms (dialogs) called frmConnect are useful for custom interfaces to connect IP addresses and ports of the client and server through sockets. An important point to note is to make sure Microsoft winsock control called `Winsock1` has been added to the form.

Arena ThisDocument Object and frmConnect UserForms codes for the model are shown in Appendix C.

6.5.2.2 Model modification and preparation

Define the specification for sending message from Arena simulation model to the external process when it is run in execution mode. This can done by changing the fixed delay time (or velocity) in the model statement blocks or modules of 'Delay', 'Route', Transport', 'Move', 'Process', 'Enter' or 'Leave' to TASKID expression which executes in real-time. The format of message string is defined in 'TASKS' element.

A message to send from a logic module or block to the external process needs to be specified. In Arena this can done by changing the fixed delay time (or velocity) in the model statement blocks or modules of 'Delay', 'Route', Transport', 'Move', 'Process', 'Enter' or 'Leave' to

TASKID (Value, TaskID [, TimeOutInterval][, ErrorLabel])

The logic of a TASKID expression is executed as two threads in parallel. The first thread simulates the delay or transfer time using the specified Value. The second thread executes the real-time task by sending a message to the real system to start an activity; it then waits for the system to respond with a "task completed" message. If the execution thread finishes before the simulation thread, the simulation thread is terminated and the entity departs the block. If the simulation thread finishes first, the entity remains suspended in the block until either (a) the execution thread completes, or (b) the actual task time exceeds Value by an amount that is greater-than-or-equal to

the TimeoutInterval, in which case the task is terminated with a timeout error and the entity is sent to the block specified by ErrorLabel.

This involves defining the format of messages the simulation entities may send to the external process. In Arena this is called experiment statements defined in 'Tasks' Element.

The TASKS element defines message strings that simulation entities may send to an external process when Arena is running in execution mode. After sending a message, an entity can then wait for a response back from the external application before proceeding to the next block. This allows us to coordinate the simulation logic with the external process of a real system.

Table 6.2 shows the modification on the Arena modules, SIMAN blocks and elements that were necessary for the simulation-emulation conversion. The changes indicated are for the following activities:

1. loading of raw parts at the arrival station on to the transporter,
2. unloading of raw parts from the transporter on to the machining centre,
3. machining process at machining centre.

Changes on other sections of the model follow similar procedure.

Table 6.2 Parameter setting differences between simulation and emulation using Arena RT.

Module Name/Variable	Simulation	Emulation
Delay Loading Arrival Station		
Delay Time	UNIF(.5,1.5)	TASKID(UNIF(.5,1,1.5), LoadingTime,NOWAIT)
Units	minutes	minutes
Delay Unload Machine Centre		
Delay Time	UNIF(0.5,1.5)	TASKID(UNIF(.5,1.5),ProcessPart))
Units	minutes	minutes
Process Machining		
Delay Time	UNIF(0.5,1.5)	TASKID(UNIF(.5,1.5), MachinePart)
Units	minutes	minutes
Tasks : LoadingTime	Not Used	
Format		"LoadPart TGID=%1.0f TNOW=%5.2f;"
Parameter		IDENT, TNOW
Tasks : UnloadingTime	Not Used	
Format		"UnloadPart TGID=%1.0f TNOW=%5.2f;"
Parameter		IDENT, TNOW
Tasks : MachinePart	Not Used	
Format		"MachinePart TGID=%1.0f TNOW=%5.2f;"
Parameter		IDENT, TNOW

Figure 6.7 shows the changes in the logic construct. It also shows the inclusion of TASKS elements for loading and unloading time as well as machine part processing real time expressions. The full emulation model can be viewed in Appendix D.

An important lesson learnt while verifying the emulation model was that TASKID expression could not be implemented for the REQUEST construct. Apparently, this is one of the modules that Arena software has not implemented the real time facility. Debugging to get the emulation model running has resulted in the use of ALLOCATE and MOVE combination from Block Panel instead.

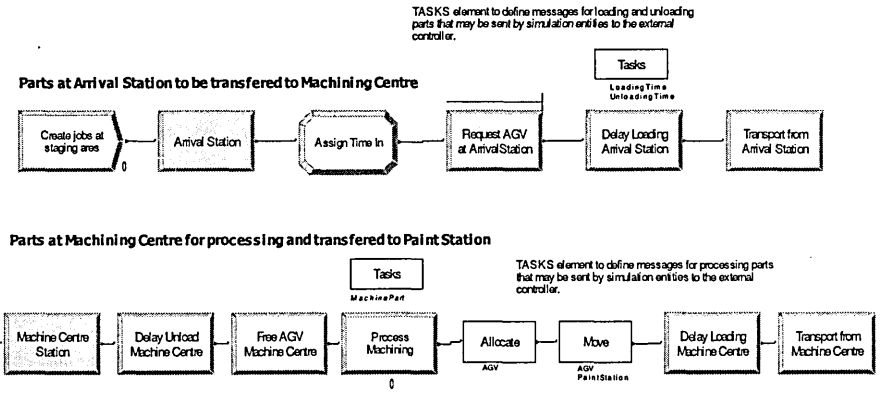


Figure 6.7 Emulation model logic at part arrival station and machining centre.

This means that at certain stage of the development, modelling HSEM in Arena sometime may require not just changing to TASKID expression in the "Delay" or "Velocity" variables but may also require substituting building blocks. In this situation, the modeller needs to be aware of the limitations of built-in objects or modules and the availability of alternative approaches and objects including the use of lower level modelling blocks or language.

Detail modelling of transporter movement requires the use of set of features for guided vehicle instead of free path which in Arena is at SIMAN blocks and elements level.

These are not reported in the thesis but investigation has shown that to get the detail model running, the set of "REQUEST", "DELAY", TRANSPORT", "FREE" and "MOVE" modules from the Basic Process and Advanced Transfer Panels have to be substituted with "Request", "Delay", Transport", "Free" and "Move" SIMAN blocks from Blocks Panel. The use of Guided Vehicle set of features necessitate the use of "Transporters", "Intersections", "Links" and "Networks" elements from the Element panel instead of "Transporter" and "Distance" data modules in Advanced Transfer Panel.

Further investigation on detailing the model to include Automated Guided Vehicle (AGV) control features like bottleneck detection, deadlock detection, collision avoidance, dynamic traffic scheduling and routing have highlighted the importance of space awareness in simulation model and uncovered the limitation of Arena of lacking it.

This has raised a new issue of synchronization of space in emulation model. So far this thesis has looked at synchronization of time as the prime requirement for the interaction between the model and controller. Synchronization of space would be easier modelled using software packages providing 3D true to scale modelling like Automod and Quest as well as Object-oriented simulation packages like eM-Plant and FlexSim. Therefore HSEM modeller needs to be aware if space awareness is required and the choice of software packages that provide the facilities.

6.5.2.3 Interface development and use

Develop and use a message handling program, an interface for sending message received from Arena to the external process and sending message received from the external process to Arena model. In this model, the message handling program interface written and used, as shown in Figure 6.7, is called RTConsole. VBA userforms "listen" and "connect" were also developed to facilitate connection between model and RTConsole.

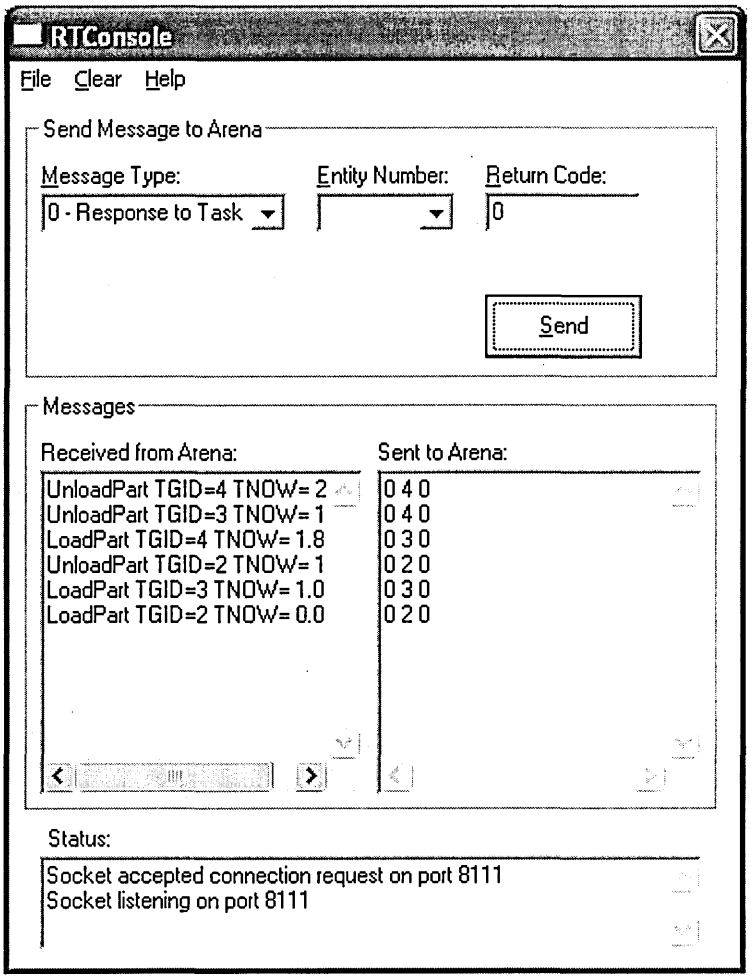


Figure 6.8 Message Handling Interface RTCosole

The program written in Visual Basic 6.0 can be viewed in Appendix E.

6.6 Model Execution

The emulation model in Arena set to run with RTConsole interface was executed according to the following procedures.

1. Set the model to run in real time mode.

a) In <Run Setup\Run Control>, verify that the model is set to "Run in Execution Mode".

b) In <Run\Setup\Run Control>, verify that " .DLL" is NOT checked unless a DLL is to be used.

c) In <Run Setup\Run Speed>, verify that the model will "Advance Simulation Time Using a Real Time Factor" of 1.

2. Invoke 'Listen' on the message handler through its local port.

a) Click the icon "RTConsole.exe" in the named view "Model Description and Instructions" of the model. This starts the client application that will send and receive messages from Arena.

b) From RTConsole's menu bar, click <File\Listen...>. Display form is as shown in Figure 6.9. Keep the default of "8111" for the port and hit <OK>. RTConsole is now waiting to connect to Arena.

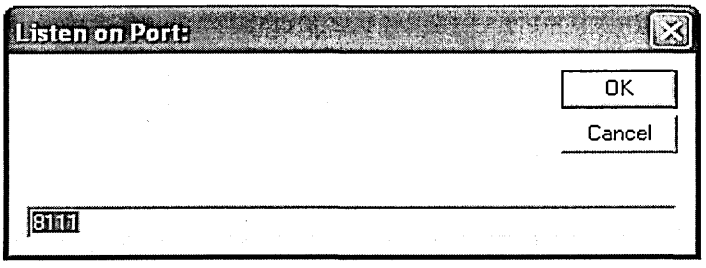


Figure 6.9 Display form for invoking 'Listen'.

3. Confirm connection to the server IP address and port.

a) From Arena's menu bar, click <Run\Go> and begin running MfgPlant_RT_AGV.doe. Note that the VBA event RealTimeInitialize is called first, displaying a dialog asking for RTConsole's IP address and port.

b) Display form is as shown in Figure 6.10. Keep the defaults and press <Connect>

c) The model should display a "Successfully connected to RTConsole" message that indicates a successful connection. Press <OK>.

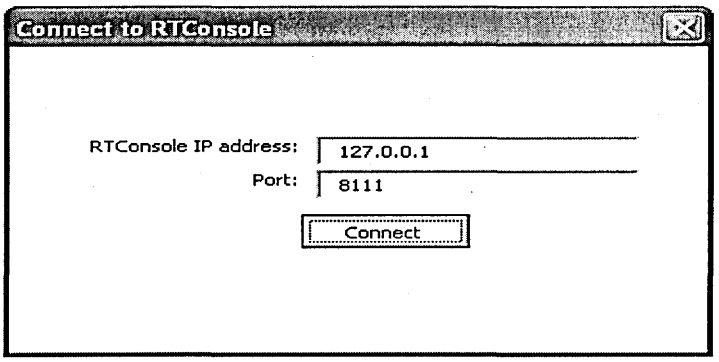


Figure 6.10 Display Form to Confirm Connection

4. The model run was successfully run with the model and the controller located on different computers. The interaction between the controller and model was done through the message handler interface in real time. Actual loading and unloading time as well as the processing time which form part of the actual WIP studies were determined by the reaction of the respective operators which acted as controllers.

Instructions for running the validation model called MfgPlant_RT_AGV.doe using VBA real time events can be viewed in Appendix F.

Table 6.3 illustrates the different hierarchy levels of the modules or panels in Arena used for three phases of simulation-emulation model development.

At present there are no specific module or block in Arena that could facilitate integration of model with real time controller. Therefore knowledge on programming and use of DLLs is essential to develop interfaces with emulation model.

Table 6.3 Panels used in development of HSEM

Basic Simulation ⇨	Detailed Simulation ⇨	Emulation
• Basic Process • Advanced Process • Advanced Transfer	• Basic Process • Advanced Process • Advanced Transfer • Blocks (SIMAN) • Elements (SIMAN)	• Basic Process • Advanced Process • Advanced Transfer • Blocks (SIMAN) • Elements (SIMAN) • User-written Visual Basic, C++ code

6.7 Summary

This chapter has described the validation of the HSEM methodology by reporting the development of an emulation model of a hypothetical manufacturing plant. The three phase starts with a conceptual base simulation model, developing into detailed model, followed with the integration with controllers in real time to complete the emulation model building.

It is acknowledged that modelling approach may vary among model builders depending on several factors including the features and structure of the simulation software package as well as the modeller's skill and experience. However the validation process has further highlighted the importance of accessibility to source codes and the knowledge to use them to model the appropriate details in the HSEM development.

A considerable amount of time and effort was needed to develop the interface to provide communication between the emulation model and the human controller. Thus the experience in the validation process has emphasised the need for special facilities and features for inter-process communication to be incorporated in the simulation software package.

Synchronization of time is seen to be the main requirement for the interactive communication between emulation model and the controller. It is also found that there are situations where synchronization of place is required. This is suggested to be another area for future research.

The next chapter presents the overall findings of the research as well recommendations for further development.

CHAPTER 7

CONCLUSIONS

7.1 Research Background

This research work started with investigating the role of simulation in manufacturing system development particularly towards shortening design to manufacture cycle time. The advantage for using simulation is that simulation can often capture and describe the complex interactions within a particular manufacturing system accurately where analytical methods seen to have failed.

As flexible and agile manufacturing become prevalent and the ability to describe short term future performance of manufacturing system becomes critical, the use of simulation has become more significant than ever.

An application of simulation that has attracted attention among manufacturing industries is for testing of control logic before installing on site. The aim is to avoid full testing using real manufacturing system and real control system which not only expensive to build and experiment but also involves high risk of failure if the possibilities of design are not tested thoroughly beforehand.

The use of full or pure simulation for control system testing using may not involve high costs but it may disregard some phenomena that are present in the real system or contain additional factors that might influence the outcomes. Thus the results may not be realistic and reliable.

Reality in the loop (RIL) testing or test bench testing involving the use of simulated control system with real equipment has been widely used on relatively small system and equipment. For large system like manufacturing this type of testing may not be practical.

Emulation or soft commissioning uses detailed simulation called emulation model of the equipment or system to be controlled with real control systems. This application has been proven to have reduced the commissioning time as well as improved efficiency and reliability of the manufacturing system delivery.

Even though the success of using emulation particularly in improving cost-effectiveness of automated material handling system delivery has been acknowledged by industries and simulation model developers, the uptake for this technology is still low.

The main reason is that it requires huge amount of initial investment in terms of cost, resource and multidisciplinary expertise because of the complex nature of its model building. The complexity is usually attributed to the model multiple level of detail, incremental development of the control logic to be tested and the inclusion of real time communication between the model and controller. Thus an emulation model is being built totally different and separated from the initial simulation model. In other words, simulation and emulation models are not convertible.

7.2 Contributions to Knowledge

The main theme of this research is to establish a methodology to develop simulation model that can be converted into emulation model with ease. The beneficiaries to the methodology would be the modellers, researchers, simulation software developers and the model client particularly the small and medium size companies where emulation technology would become more affordable. The contributions are stated in brackets at the end of the appropriate paragraph.

7.2.1 HSEM Methodology

The main contribution of the research to new knowledge is the Hybrid Simulation Emulation Model (HSEM). HSEM is a product and a new approach of developing emulation model that would help reduce the negative perception amongst modellers and clients towards emulation of being difficult to develop, costly and non-productive. Consequently, a more positive perception and clearer model building approach would increase the readiness among small companies to invest in this technology. **(Contribution 1)**

The novel methodology to develop new product of HSEM presented in Chapter 5 is a synthesis and extension of many ideas and developments covering the topics of manufacturing system development, methodology for modelling details, real time simulation methodology as well as verification and validation of simulation model. In short the methodology comprises of three sequential steps as follows.

1) Development of base simulation model,
2) Development of detail emulation model,
3) Integration of controller with the emulation model.

As shown in Chapter 5, the approach is developmental and iterative. To implement the methodology, like any simulation project, HSEM modelling requires a team of appropriate expertise, knowledge of simulation methodology and a selection of the right simulation tools. The product of this research work in the form of a methodology has taken into consideration those requirements above. The organization of the methodology would be useful in assisting the HSEM project manager to organise and monitor the project effectively. **(Contribution 2)**

7.2.2 Facilitating decision making process

HSEM approach which is developmental and incremental would facilitate the development of integrated decision support system at all three levels of managerial planning, namely strategic, tactical and operational levels. For example, one of the decision making areas at the corporate level can be design of a manufacturing system. After a broad architecture of the system is identified, a HSEM may be developed to answer the detailed design issues like number of machines required, the type of computer control at each level etc. Several design alternatives can be evaluated and the one best suits the business goal can be selected.

At a tactical level, the impact of a decision taken at strategic level can be emulated and the output can be analysed to get more insight into the system behaviour. For example, for a particular design configuration chosen, the material requirement plan over the planning horizon can be emulated. Similarly emulation of inventory systems helps finding best inventory policy for the system. At this level, a detailed model may even be used to compare the projected work load generated from MRP with machine and labour resources available. The model may assist the process of capacity management by producing profiles of load against capacity over time. This in turn assists in management decision-making regarding short-term capacity planning and adjustment.

At operational level, HSEM may be used for a detailed, day-to-day scheduling, which is derived from production plans made at higher levels. At this level, it may also be used as a part of the overall shop floor control. The current status of the shop floor can be maintained using an interface with the shop floor software as well as the factory database. Depending on the current status, various production control decisions can be specified. The effect of such decisions on the progress of the orders in terms of the due dates, WIP, and machine utilization level could then be checked. The type of decision to be emulated would include, for example, the effect of batch sizes and times of batch release. The progress of jobs through a shop can be emulated

once a particular control decision has been implemented. From the results, proper adjustments could be made in the schedule.

Decisions at various levels of a system highly influence each other. The consequence of ignoring such real-time interaction could be very serious. Using HSEM, managers can analyse various decisions at any planning level, and observe the effects of these decisions on the other levels. **(Contribution 3)**

7.2.3 Another perspective of emulation technology

Emulation which is considered to be an extension of simulation technology is in a situation similar to when simulation was emerging and perceived to be only for large companies. As reported in Chapter 2, publications on emulation are limited. Thus not many know about its technology and the cost-benefit it brings. This thesis provides better awareness as well as another perspective of emulation technology and hope to attract the attention of wider spectrum of potential users. **(Contribution 4)**

The major inhibitors to widespread use of modelling and simulation in manufacturing are the perception of its high costs of model building and low reuse. The perception is even worse for emulation due to the fact that it being more detailed and complex. Section 2.4 of Chapter 2 explains simulation-emulation relationships while section 2.5 reports some application of emulation. This would provide better understanding of emulation technology and with that the modeller and client would be able to make a more informed decision with regard to investing in emulation technology.
(Contribution 5)

7.2.3 Potential areas of development

The results of the questionnaire survey conducted in this research work, as reported in Chapter 3, have indicated increasing interest among simulation users in the use of emulation technology as well as have identified priority areas of development. This

would help future researchers justify and prioritise their work, especially in developing tools and facilities for building emulation model easier and more user-friendly. **(Contribution 6)**

Chapter 4 describes the requirements and criteria of functionalities in simulation software packages that are needed for the development of HSEM. They are categorised into features that should provide (1) the flexibility of adding details to the simulation model while assuring its correctness and (2) the inter process communication between model and real control system.

The specific emulation modelling features and the comparison of features for HSEM against selected simulation software packages presented in section 4.4 *could* be used by the HSEM developer as benchmark for selecting the appropriate software package for the project. To the simulation software vendor, these would help to identify specific areas for feature enhancement or tool development. **(Contribution 7)**

7.2.4 Convertible Simulation-emulation model

As reported in Chapter 6, the research work has proved that HSEM is viable, refuting the perception that simulation and emulation are non convertible. This would pave the way for more aggressive work by future researchers on making emulation model more affordable. **(Contribution 8)**

As a result of emulation technology in the form of HSEM becoming more affordable, the industry in general would benefit in terms of low overall cost, efficient use of resource and client satisfaction. Also, the consumers at large would be able to receive better and faster product delivery as well as wider choice of products. **(Contribution 9)**

7.2.5 Summary of Contributions to Knowledge

The contributions of this research to knowledge is summarised according to the benefits classified in the previous sections and their beneficiaries shown in Table 7.1.

Table 7.1 Summary of Contributions to Knowledge

	Beneficiaries			
Contributions And Benefits	Modellers	Software developers and researchers	Client (Industry)	Product Customer (Public)
1	◙		◙	
2	◙			
3	◙		◙	
4	◙		◙	
5	◙		◙	
6		◙		
7	◙	◙		
8		◙	◙	
9			◙	◙

7.3 Recommendations for future work

The present work has produced a generic methodology and guide for developing HSEM. Thus it is open to modellers to develop models in their own application area using their choice of simulation software. However below are some suggestions for future researchers to work on to make emulation modelling more appealing and useful.

7.3.1 Enriching existing special packages.

Although the focus of the current work is on the application of HSEM in manufacturing system, there are also other potential application areas that could use HSEM methodology. As reported in Chapter 3, those areas include transport system, business process and security system. Some software vendors offer special packages for example to model logistics, healthcare and call centre which include templates and functionalities for specific applications. Blending these functionalities with real time (RT) modelling tools would significantly reduce the effort to develop emulation model in the respective area.

However, compatibility of certain modules or blocks with RT modeling tools needs to be investigated or perhaps modified before use. For example in the context of modelling using Arena, as indicated in section 6.5.2.2, while Arena is equipped with RT modelling tools there are modules and blocks that are presently not compatible. These could be developed further either by the software developer or the modeller to enhance the use of RT modelling tool for emulation modelling.

7.3.2 Interface development tools

The experience of developing the method and validating the methodology, as reported in Chapter 6 has highlighted the need to have access to lower level codes and the skill to use them in modelling detail. At present some level of programming skill is required to program and use the appropriate tools or functions to develop system interface for inter process communication. It is widely accepted fact that easy model building means requiring no or minimal programming. Thus this work has emphasised the need for future researchers and software developers alike to develop more user friendly codes or blocks for developing inter process communication interface. This would be useful to enable the emulation modeller to concentrate on developing emulation model which itself is already complex.

7.3.3 More Comprehensive Feature Review

It is acknowledged that enhancements and new functionalities are continuously being developed by the simulation software vendors and new features are introduced in version upgrades. Thus it is difficult for the researcher to make authoritative comparisons on the availability and accessibility of certain functions in the software packages in the market. The software comparison for HSEM suitability presented in Chapter 4 can be used as a guide and researchers as well as HSEM modellers are recommended to regularly review and update the comparison table so as to make the comparison more comprehensive and up-to-date.

7.4 Conclusions

This research has devised and developed a new approach to develop emulation model called Hybrid Simulation Emulation Model (HSEM). Prior to this, simulation model and emulation model are considered to be non convertible. This work has proved that, by applying the methodology devised and described in the thesis, a simulation model built for initial analysis and development can be converted into an emulation model for testing a control system.

Potentially the methodology developed offer a starting point for further research into the comprehensive support for the implementation of real time control system testing using emulation. Hence the future application of this methodology and recommendations may lead to important technical and commercial benefits.

REFERENCES

Aguilar-Savén, R. S. R. S. (2004). "Business process modelling: Review and framework." International Journal of Production Economics **90**(2): 129-149.

Auinger, F., M. Vorderwinkler and G. Buchtela (1999). Interface Driven Domain-Independent Modeling Architecture For "Soft-Commissioning" And "Reality In The Loop". The 1999 Winter Simulation Conference, Phoenix, AZ, USA.

Balci, O. (2003). Verification, Validation, And Certification Of Modeling And Simulation Applications. The 2003 Winter Simulation Conference, New Orleans, LA, USA.

Ball, P. (1998). "Abstracting Performance In hierarchical Manufacturing Simulation." Journal Of Materials Processing Technology **76**(1-3): 246-251.

Ball, P. D. (1996). Introduction to discrete event simulation. the 2nd DYCOMANS workshop on "Management and Control : Tools in Action", Algarve, Portugal,.

Banks, J. (1998). Handbook of simulation : principles, methodology, advances, applications and practice. New York, Wiley : Engineering and management press.

Banks, J. (2000). Simulation In The Future. The 2000 Winter Simulation Conference, Orlando, FL, USA.

Banks, J., J. S. Carson, B. L. Nelson and D. M. Nicol (2000). Discrete event system simulation. Upper Saddle River, New Jersey, USA, Prentice Hall.

Bapat, V. and D. T. Sturrock (2003). The Arena Product Family: Enterprise Modeling Solutions. The 2003 Winter Simulation Conference, New Orleans, LA, USA.

Benjamin, P., M. Erraguntls, D. Delen and R. Mayer (1998). Simulation Modelling at Multiple Levels of Abstraction. The 1998 Winter Simulation Conference, Washington DC, USA.

Boer, C. A., A. Verbraeck and H. P. M. Veeke (2002). The Possible Role Of A Backbone Architecture In Real-Time Control And Emulation. The 2002 Winter Simulation Conference, Salt Lake City, UT, USA.

Brennan, R. W. (2000). " Performance comparison and analysis of reactive and planning-based control architectures for manufacturing." Robotics and Computer Integrated Manufacturing **16**(2-3): 191-200.

Brennan, R. W. and D. H. Norrie (2001). "Evaluating The Performance Of Reactive Control Architectures For Manufacturing Production Control." Computers In Industry **46**(3): 235-245.

Cheshire, C. and J. Hodgson (2001). Using AutoMod and Emulation, Coca-cola improved system throughput by 17%. The 2001 Brooks Automation Symposium Proceedings.

Dalal, M., B. Groel and A. Prieditis (2003). Real-Time Decision Making Using Simulation. The 2003 Winter Simulation Conference, New Orleans, LA, USA.

Davis, W. J., J. G. Macro, A. L. Brook, M. S. Lee and G. S. Zhou (1996). Developing A Real-Time Emulation/Simulation Capability For The Control Architecture To The Ramp FMS. The 1996 Winter Simulation Conference, Coronado, CA, USA.

Dougall, D. J. (1998). Applications and benefits of real-time I/O simulation for PLC and PC control systems. ISA Transactions. **Vol.36:** 305-311.

Eldabi, T. and R. J. Paul (1997). Flexible Modeling Of Manufacturing Systems With Variable Levels Of Detail. The 1997 Winter Simulation Conference, Atlanta, GA, USA.

Gonzalez, F. G. and W. J. Davis (1997). A Simulation-Based Controller For Distributed Discrete-Event Systems With Application To Flexible Manufacturing. The 1997 Winter Simulation Conference, Atlanta, GA, USA.

Gonzalez, F. G. and W. J. Davis (1998). Developing A Physical Emulator For A Flexible Manufacturing System. IEEE International Conference on Systems, Man and Cybernetics.

Habchi, G. and C. Berchet (2003). "A model for manufacturing systems simulation with a control dimension." Simulation Modelling Practice and Theory **11**(1): 21-44.

Harrell, C. R. and R. N. Price (2002). Simulation Modeling Using Promodel Technology. The 2002 Winter Simulation Conference, Salt Lake City, UT, USA.

Hasnan, K. and T. Perera (2004). Hybrid Simulation-Emulation Model: AGVS Example. ESDA04 7th Biennial Conference On Engineering Systems Design And Analysis, Manchester, United Kingdom, ASME.

Hasnan, K., T. Perera and D. Clegg (2005). User Perspectives On The Use Of Emulation In Control System Testing. 3rd International Conference on Manufacturing Research (ICMR 2005), Cranfield University, UK.

Heinicke, M. U. and A. Hickman (2000). Eliminate Bottlenecks with Integrated Analysis Tools in eM-Plant. The 2000 Winter Simulation Conference, Orlando, FL, USA.

Hlupic, V. (2000). Simulation Software: An Operational Research Society Survey Of Academic And Industrial Users. The 2000 Winter Simulation Conference, Orlando, FL, USA.

Hodgson, J. and M. Kartz (2000). Using A Portable Simulation Structure With Emulation For Offline Testing, AutoSimulations Symposium. AutoSimulations Symposium.

Holst, L. (2001). Integrating Discrete-Event Simulation into the Manufacturing System Development Process : A Methodological Framework. Division of Robotics, Department of Mechanical Engineering. Lund, Sweden, Lund University.

Horton, I. (1998). Beginning Visual C++ 6. Birmingham, Wrox Press.

Hunter, M. and R. Machemehl (2003). Development And Validation Of A Flexible, Open Architecture, Transportation Simulation. The 2003 Winter Simulation Conference, New Orleans, LA, USA.

Jeong, K.-C. and Y.-D. Kim (1998). "Real-time scheduling mechanism for a flexible manufacturing system: Using simulation and dispatching rules." International Journal of Production Research **36**(9): 2609-2626.

Joines, J. A. and S. D. Roberts (1999). Simulation In An Object-Oriented World. The 1999 Winter Simulation Conference, Phoenix, AZ, USA.

Jordan, S. E., M. K. Snell, M. M. Madsen, J. S. Smith and B. A. Peters (1998). Discrete-Event Simulation For The Design And Evaluation Of Physical Protection Systems. The 1998 Winter Simulation Conference, Washington DC, USA.

Julia, S. and R. Valette (2000). "Real time scheduling of batch systems." Simulation Practice and Theory **8**(5): 307 - 319.

Kelton, W. D., R. P. Sadowski and D. T. Sturrock (2004). Simulation With Arena, Third Edition. New York, McGraw Hill.

Klingstam, P. and P. Gullander (1999). "Overview of simulation tools for computer-aided production engineering." Computers in Industry **38**(2): 173-186.

Krahl, D. (2003). Extend: An Interactive Simulation Tool. The 2003 Winter Simulation Conference, New Orleans, LA, USA.

Law, A. M. and W. D. Kelton (2000). Simulation modeling and analysis. New York, McGraw-Hill.

Law, A. M. and M. G. Mccomas (1999). Simulation Of Manufacturing Systems. 1999 Winter Simulation Conference, Phoenix, AZ, USA.

LeBaron, H. T. and R. A. Hendrickson (2001). Using Emulation To Validate A Cluster Tool Simulation Model. The 2001 Winter Simulation Conference, Arlington, VA, USA.

LeBaron, T. E. and K.Thompson (1998). Emulation Of A Material Delivery System. The 1998 Winter Simulation Conference, Washington DC, USA.

Lee, K. and P. A. Fishwick (1999). "OOPM/RT: A Multimodeling Methodology For Real-Time Simulation." ACM Transactions On Modeling And Computer Simulation **9**(2): 141-170.

Mcgregor, I. (2000). The Use Of Emulation In Commissioning: An Example Model. AUTOFLASH. **13**.

Mcgregor, I. (2002). The Relationship Between Simulation And Emulation. The 2002 Winter Simulation Conference, Salt Lake City, UT, USA.

Mehrabi, M. G., A. G. Ulsoy and Y. Koren (2000). "Reconfigurable Manufacturing Systems: Key To Future Manufacturing,." Journal Of Intelligent Manufacturing **11**: 403 - 419.

Meinert, T. S., G. D. Taylor and J. R. English (1999). " A modular simulation approach for automated material handling systems,." Journal of Simulation Practice and Theory **7**: 15-30.

Mueller, G. (2001). Using Emulation To Reduce Commissioning Costs On A High Speed Bottling Line. The 2001 Winter Simulation Conference, Arlington, VA, USA.

Nikoukaran, J. and R. J. Paul (1999). "Software selection for simulation in manufacturing: a review." Simulation Practice and Theory **7**(1): 1-14.

Nketsa, A. and R. Valette (2001). "Rapid and Modular Prototyping-based Pet Nets and Distributed Simulation for Manufacturing Systems." Journal of Applied Mathematics and Computation: 265-278.

Nordgren, W. B. (2003). Flexsim Simulation Environment. The 2003 Winter Simulation Conference, New Orleans, LA, USA.

Oppenheim, A. N. (1992). Questionnaire design, interviewing and attitude measurement. London, Pinter Pub Ltd.

Page, E. H. and R. Smith (1998). Introduction To Military Training Simulation: A Guide For Discrete Event Simulationists. The 1998 Winter Simulation Conference, Washington DC.

Persson, J. F. (2002). "The impact of different levels of detail in manufacturing systems simulation models." Journal of Robotics and Computer Integrated Manufacturing **18**: 319-325.

Peters, B. A., J. S. Smith, J. Curry and C. LaJimodiere (1996). Advanced Tutorial - Simulation-based Scheduling and Control, Proceedings Of. The 1996 Winter Simulation Conference, Coronado, CA, USA.

Pidd, M. and R. B. Castro (1998). Hierarchical Modular Modelling In Discrete Simulation. The 1998 Winter Simulation Conference, Washington DC, USA.

Rabbath, C. A., M. Abdoune and J. Belanger (2000). Effective Real-Time Simulations Of Event-Based Systems. The 2000 Winter Simulation Conference, Orlando, FL, USA.

Randell, L. G., L. G. Holst and G. S. Bolmsj¨o (1999). Incremental System Development Of Large Discrete-Event Simulation Models. The 1999 Winter Simulation Conference, Phoenix, AZ, USA.

Rawles, I. (1998). The Witness® Toolbox - A Tutorial. The 1998 Winter Simulation Conference, Washington DC, USA.

Rengelink, W. and Y. A. Saanen (2002). Improving The Quality Of Controls And Reducing Costs For On-Site Adjustments With Emulation:An Example Of Emulation In Baggage Handling. The 2002 Winter Simulation Conference, Salt Lake City, UT, USA.

Robertson, N. and T. Perera (2002). "Automated data collection for simulation?" Simulation Practice and Theory **9**(6-8): 349-364.

Robinson, S., R. E. Nance, R. J. Paul, M. Pidd and S. J. E. Taylor (2004). "Simulation model reuse: definitions, benefits and obstacles." Simulation Modelling Practice and Theory **12**(7-8): 479-494.

Rogers, P. and R. W. Brennan (1997). A Simulation Testbed For Comparing The Performance Of Alternative Control Architectures. The 1997 Winter Simulation Conference, Atlanta, GA, USA.

Rohrer, M. and I. W. McGregor (2002). Simulating Reality Using AutoMod. The 2002 Winter Simulation Conference, Salt Lake City, UT, USA.

Rumbaugh, J. (1991). Object-oriented Modelling and Design. New Jersey, Prentice Hall International.

Sadowski, D. A. and M. R. Grabau (1999). Tips For Successful Practice Of Simulation. The 1999 Winter Simulation Conference, Phoenix, AZ, USA.

Schiess, C. (2001). Emulation : Debug in the Lab - Not On The Floor. The 2001 Winter Simulation Conference, Arlington, VA, USA.

Schludermann, H., T. Kirchmair and M. Vorderwinkler (2000). Soft-Commissioning: Hardware-In-The-Loop-Based Verification Of Controller Software. The 2000 Winter Simulation Conference, Orlando, FL, USA.

Shnits, B., J. Rubinovitz and D. Sinreich (2004). "Multicriteria dynamic scheduling methodology for controlling a flexible manufacturing system." International Journal of Production Research **42**(17): 3457–3472.

Smith, J. S. (2003). "Survey on the use of simulation for manufacturing system design and operation." Journal of Manufacturing Systems **22**(2): 157.

Smith, J. S., B. A. Peters, S. E. Jordan and M. K. Snell (1999). Distributed Real-Time Simulation For Intruder Detection System Analysis. The 1999 Winter Simulation Conference, Phoenix, AZ, USA.

Stewart, P., P. J. Fleming and S. A. MacKenzie (2003). "Real-time simulation and control systems design by the response surface methodology and designed experiments." International Journal of Systems Science **34**(14-15): 837-850.

Thomas, S. J. (1999). Designing surveys that work! : a step-by-step guide. Thousand Oaks, California, Corwin Press.

Vedapudi, S. (2001). Using MCM To Do Emulation Of A Car Assembly Line. The 2001 Brooks Automation Symposium Proceedings.

Verbraeck, A. and C. Versteegt (2000). A bridge between the design and implementation of complex transportation systems - Linking simulation models and physical models. 12th European Simulation Symposium ESS2000 - Simulation in Industry, Hamburg, Germany, SCS publications,Ghent.

Verbraeck, A. and C. Versteegt (2001). Logistic for fully automated large-scale freight transport systems. 2001 IEEE Intelligent Transportation Systems Conference, Oakland (CA), USA.

Versteegt, C. and A. Verbraeck (2002). The Extended Use Of Simulation In Evaluating Real-Time Control Systems Of AGVS And Automated Material Handling Systems. The 2002 Winter Simulation Conference, Salt Lake City, UT, USA.

Wells, G. J., R. E. Zee and C. J. Damaren (2002). " Hardware Emulation Strategies for Concurrent Microsatellite Hardware and Software Development." The Canada Aeronautics and Space Journal **48**(1): 87-95.

Wright, P. (1998). Beginning Visual Basic 6 Objects. Birmingham, U.K., Wrox Press.

Wu, B. (1994). Manufacturing systems design and analysis : context and techniques. London, Chapman and Hall.

Xu, J., K. L. Hancock and F. Southworth (2003). Dynamic Freight Traffic Simulation Providing Real-Time Information. The 2003 Winter Simulation Conference.

Young, J. and W. Heider (2002). Emulating An Order Fulfillment System. Brooks-PRI Automation Simulation Symposium 2002, Salt Lake City, UT, USA.

Appendix A

Survey Questionnaire

Control System Testing Using Simulation Survey

Although **control system testing using simulation** also known as '**emulation**' has been used extensively in the design on manufacturing systems, it would also applicable in other areas as well. Examples are logistic control in transportation and defence, production process control as well as business process control.

The aim of the survey is to obtain feedback from simulation practitioners on the usage of simulation for control system testing, their application areas and development problems. This will then help us to identify areas of potential research to enhance the use of emulation.

If you would like further information on our work, please contact us at the School of Engineering, Sheffield Hallam University, England.

Khalid Hasnan
k.hasnan@shu.ac.uk

or

Prof Terrence Perera
t.d.perera@shu.ac.uk

==========

1. Are you an academic or industrial user of simulation software?

☐ Academic

☐ Industrial

2. Do you use simulation to test control system in your work?

☐ Yes — Please go to Question 3

☐ No — Please go to Question 6.

3. Which application area do you use simulation for control system testing? (You can tick more than one box.)

☑ Process Control (Production)

☐ Material Handling (e.g.Manufacturing)

☐ Transportation (Cargo, Passenger)

☐ Business Process

☐ Security System

☐ Other Please specify. _____

4. Please rank the following in terms of the benefit of using simulation for control system testing? (1 = Greatest, 4 = Least)

☐ Short Commissioning Time

☐ Low Overall Cost

☐ Effecient Use of Resource

☐ Client Satisfaction

☐ Other Suggestion _____

5. **In designing simulation model for testing control system, which development stage would require a specific tool or module to facilitate emulation model building? Please rank according to its importance. (1 = Greatest, 3 = Least)**

☐ Determining the correct level of detail for model representation

☐ Modifying simulation code

☐ Interfacing between models

☐ Other Please specify

6. **Which simulation package(s) do you primarily use?**

☐ Arena ☐ Automod ☐ AweSim

☐ EmPlant/Simple++ ☐ Extend ☐ Promodel

☐ Quest ☐ Witness

☐ Other Please specify

7. **Which application area do you think would benefit from the advancement of control system testing using simulation technology?** (You can tick more than one box.)

☐ Process Control (Production) ☐ Business Process

☐ Material Handling (e.g. Manufacturing) ☐ Security System

☐ Transportation (Cargo, Passenger)

☐ Other Please specify.

Please feel free to add any comments about anything you feel is particularly important to facilitate modifying a simulation model into a model that is capable to test a real control system.

our e-mail address

Error! Objects cannot be crea **(Optional** - please enter your e-mail address if you would be interested in sharing of information in this area and/or

Thank you.

Appendix B

Conference Paper Proceedings

Appendix B.1	Hybrid Simulation-Emulation Model: AGVS Example. (ESDA 2004, Manchester UK)
Appendix B.2	User Perspectives on the Use of Emulation in Control System Testing. (ICMR 2005, Cranfield UK)

ESDA2004-58130

HYBRID SIMULATION-EMULATION MODEL : AGVS EXAMPLE.

Khalid Hasnan
Terence Perera

School of Engineering
Sheffield Hallam University
Sheffield S1 1WB
United Kingdom

Keywords : Emulation, Simulation, Arena RT, AGV.

ABSTRACT

Simulation and emulation have several salient contrasting functions and features. They include different aims, levels of details, execution time and integration of models.

In many cases, a project will require both a simulation model for initial analysis and development, as well as an emulation model for testing a control system. If this is the case, a copy of the simulation model can be used as a starting point for developing the emulation model. Hybrid simulation-emulation model, one that is used for both purposes should have a facility to switch off/on certain elements from the model as necessary.

There is much published work in simulation and a dearth of work in emulation. To date there has been no work published in converting a simulation into emulation model.

This paper describes a novel approach which combines both attributes and is illustrated using a case study based on an Automated Guided Vehicle System (AGVS).

1 Introduction

The current trend towards highly automated systems that are intended to adapt quickly to change while providing extensibility through a modular, distributed design. **[1, 2]**

In order to realise the flexibility and productivity that these advanced system promise, system modelling, simulation and control are viewed as increasingly vital to enable the components of these automated manufacturing systems to work together in an integrated way.

A growing application of simulation and control is emulation, where a simulation model is used to replace a real Automated Material Handling System (AMHS) in order to test and debug an industrial control system.

Emulation provides a reliable and safe way of verifying control code functionality offline, training operators in a safe environment, and of testing modifications to a control system before they get put into effect. The emulation model can also serve as a test bed for any further control system modifications throughout the life of the system, so production is not disrupted.

Designing an emulation model is similar to designing a simulation model; but there are some important differences [3]

- **An emulation model must include communication logic.** Emulation models open connections with one or more controllers for example OPC servers, and read and write item values throughout a model run.
- **An emulation model is often more detailed than a simulation model.** Because the emulation model must provide the same responses to the controllers as real system hardware, the model must be designed to respond to many system events that would otherwise not require custom processing during a simulation. (For example, an emulation model might be required to send signals to a controller server when a load begins a pop-up transfer, when the transfer has completed lifting, when the load moves to the new section and when the transfer completes lowering.)
- **Simulation model typically has no direct links to external devices.** Emulation model often responds to signals from the control system, which controls system processes.

Even though the benefits of using emulation for the analysis of manufacturing systems are well acknowledged, the speed and cost of its model building remains a concern.

At present, a project will require a simulation model for initial analysis and development as well as an emulation model for testing a control system. It is unlikely that the emulation model will be suitable for the initial analysis phase. The main reason is that the control system is unlikely to be available as it is usually a result of the definition stage. A second reason is that the emulation model is likely to be much more detailed than the analysis model, and so will run more slowly, making the necessary simulation analysis prohibitive. [4]

A proposed solution to this is to develop a hybrid simulation-emulation model or composable simulation model, one that is used for both purposes and should have a facility to switch off/on certain elements from the model as necessary.

In this paper we provide insight on the methodology to convert an existing simulation model into an emulation model using Arena Simulation package.

2 Manufacturing Systems and Emulation Review

Understanding the manufacturing systems entities is crucial for the success of a manufacturing emulation project where communication between entities and the elements in the manufacturing system has to be in real time. Table 1 shows this relationship, and addresses the importance of material and information transfers/flows.

Table 1 Manufacturing System Entities Relationship

	Plant		Control
Processing	Machines, material processing operations, storage buffers, machine setup, inspection	Commands ← → Status updates	Controllers, operators, machine state information, controller domains, process recipes, machine scheduling, process monitoring
	↕ Material transfer via shared locations		↕ Information transfer via networks
Transportation	Transporters, conveyors, material movement operations.	Commands ← → Status updates	Controllers, operators, transporters state information, controller domains, material movement requests, process plans

These entities are classified along two main axes: plant vs. control, and processing vs. transportation . Elements in the plant classification comprise the physical factory, such as machines, material and transporters. Elements in the control classification comprise the logical factory, including decision-makers, performance evaluators and information about the physical factory. Elements in the processing classification focus on the intermediate transformation steps that turn raw materials into finished goods, while elements in the transportation classification address the logistics of moving material through the various process stages. [5]

For the project to be cost effective, the developer has to be selective on which part of the simulation model need to be emulated. This can be achieved by reviewing the key performance indicators (KPIs) and identifying their associated variables that would have great influence. For example, in a flexible manufacturing plant a decision may require varying certain parameters in the processing and transportation variables as shown in Table 1.

3 Steps for simulation to emulation conversion

There are three basic steps in building emulation model from existing simulation mode. The steps below are generic and applicable to other simulation software packages. Arena software package was used as an example.

3.1 Model Structure

Abstract to the appropriate level of details of the control elements (modules/ process/ variables/ parameters) that are affecting the KPIs. This necessitates the model to be modular, hierarchical and configurable as shown in Fig.1 [6]

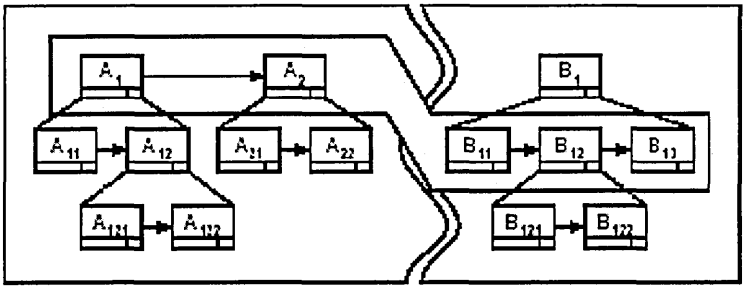


Fig. 1 Integration of different hierarchical module

Arena employs an object-oriented design for entirely graphical model development. The graphical modules can be used by simulation analysts to create models and are provided "off-the-shelf" with Arena. [7] It also provides the integration of different hierarchical Arena modules and SIMAN codes. These modules can also be custom designed to produce a modelling environment that is tailored to a specific application area, for example to facilitate emulation model building. The resulting collection of user-created modules would then be contained inside an Application Solution Template (AST) that can be shared by any licensed Arena user.

3.2 Communication

Establish the communication procedure in Real Time. Communications can be achieved through sockets, NetDDE, DCOM, or OPC. A model communication structure is shown in Fig. 2.

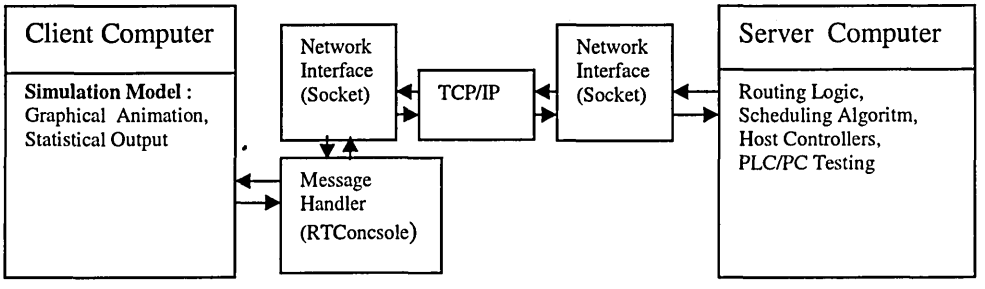


Fig 2 Emulation Model Communication Structure

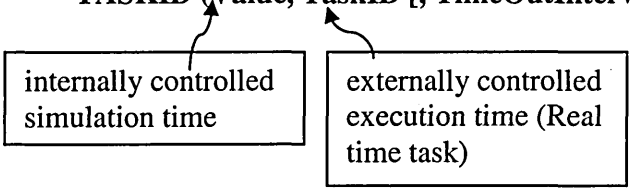
As in the case study example, models can communicate through sockets to open and close socket connections, send and read data messages (including strings and C structures), and send and read synchronization messages.

It involves three important phases:

3.2.1 Model modification and preparation

A message to send from a logic module or block to the external process needs to be specified. In Arena this can done by changing the fixed delay time (or velocity) in the model statement blocks or modules of 'Delay', 'Route', Transport', 'Move', 'Process', 'Enter' or 'Leave' to

TASKID (Value, TaskID [, TimeOutInterval][, ErrorLabel])



The logic of a TASKID expression is executed as two threads in parallel. The first thread simulates the delay or transfer time using the specified Value. The second thread executes the real-time task by sending a message to the real system to start an activity; it then waits for the system to respond with a "task completed" message. If the execution thread finishes before the simulation thread, the simulation thread is terminated and the entity departs the block. If the simulation thread finishes first, the entity remains suspended in the block until either (a) the execution thread completes, or (b) the actual task time exceeds Value by an amount that is greater-than-or-equal to the TimeoutInterval, in which case the task is terminated with a timeout error and the entity is sent to the block specified by ErrorLabel.

3.2.2 Message Handling

This involves defining the format of messages the simulation entities may send to the external process. In Arena this is called experiment statements defined in 'Tasks' Element.

The TASKS element defines message strings that simulation entities may send to an external process when Arena is running in execution mode. After sending a message, an entity can then wait for a response back from the external application before proceeding to the next block. This allows us to coordinate the simulation logic with the external process of a real system.

The ARRIVALS element creates batches of entities that arrive at the system model at specified times. In execution mode, the time is specified by key hit or message and the corresponding operands initiated by external process.

3.2.3 Inter-Process Communication

To coordinate simulation logic with the external process of a real system, he simulation model need to be programmed for the control system to open socket connections and read and send messages. In Arena this can be done using SIMAN, VBA and C++.

For example, the following VBA events provided in Arena's ThisDocument object is used. These events are only called when Arena is running in execution mode.

- ***RealTimeInitialize***—called at the beginning of the first replication. Place code that initializes the inter-process communications here.
- ***RealTimeSend***—Called when an entity tries to send a message to the external process. Place code that sends the message to the IPC queue here.
- ***RealTimeRetrieve***—Code that retrieves messages from the IPC queue and passes them to Arena here.
- ***RealTimeTerminate***—called at the end of the last replication. Place code that terminates the inter-process communications here. [6]

UserForms (dialogs) are useful for custom interfaces to connect IP addresses and ports of the client and server through sockets.

3.3. Run/Simulate the model.

- Set to run in Real Time mode (execution mode in Arena) in the Run Setup menu.
- Establish connection to simulation model (client) by invoking 'listen' on the message handler through its local port.
- Confirm connection by invoking the 'connect' method to the server IP address and port.
- Run the model.

4. Case Example

The case example is the modelling of a four manufacturing cells system. The system model also consists of part arrivals, and part departures. Cells 1, 2 and 4 each have single machine; Cell 3 has two machines. The system produces three parts, each visiting different sequence of stations. The parts are transported between stations by means of two Automated Guided Vehicles (AGV).

4.1 Model Logic Example

The Arena model in Fig. 3 shows the modelling logic and animation of the manufacturing system based on the assumed time projection of events. In reality assumed time projections are not enough. For example, due to limited buffer space the machines need to communicate between each other as well as with the arrival controller to monitor the number of parts coming to the respective stations.

The model, shown in Fig. 4, is a modification of the simulation model incorporating Real Time elements. Only the logic diagram is shown as the animation diagram in this example is the same as for simulation. This model demonstrates Arena running in execution mode and conducting inter-process communications with an external client application called RTConsole.exe written in Visual Basic.

The communication between machine computers client/server uses socket, a program device which supplies sending and receiving data via the defined TCP/IP port. In this case example, socket technology for MS Windows called Winsock was used.

The simulation of sending and receiving of the relevant parts by the respective machine on a 'client' was successfully controlled through a remote computer which acted as a 'server'.

4.2 Message Handling Example

When Arena is run in execution mode, each part will send a "LoadPart" message to the external message handler, RTConsole, before loading part on to an AGV and transporting it to the next station. Also when an AGV begins unloading on to a station, it will send an "UnloadingPart" task to RTConsole. A "task complete" response must then be sent back to Arena to indicate when the part has completed its unloading, free the AGV. The AGV then waits for or execute the next command.
As an example, simulation code for loading a part on to an AGV at a station.

REQUEST : *AGV(SDS,AGV#)*
DELAY : *UNIF(0.5,1.5)*
TRANSPORT: *AGV(AGV#)*

In the control application, the logic is identical except that the time delay depends on the performance of a physical task rather than an internal clock.

REQUEST : *AGV(SDS,AGV#)*
DELAY : *TASKID(UNIF(0.5,1.5), LoadingTime)*
TRANSPORT: *AGV(AGV#)*

The difference in the setting and code, where there are differences, between simulation and emulation models can be viewed in Table 1.

Table 2. Parameter setting differences between simulation and emulation using Arena RT.

Module or Block or Element (Name/Variable)	**Simulation**	**Emulation**
DELAY Loading To AGV# Load Time Delay Units	 UNIF(0.5,1.5) minutes	 TASKID(UNIF(0.5,1.5),LoadingTime) minutes
* **ENTER** Unloading To Cell 1 Unload Time Delay Units	 UNIF(0.6,1.3) minutes	 TASKID(UNIF(0.6,1.3),UnLoadingTime) minutes
TASKS Loading Time Format Parameter	 Not Used	 "LoadPart %1.0f TGID=%1.0f loc %1.0f TNOW=%5.2f" Entity.Type, IDENT, Entity.Station, TNOW
TASKS Unloading Time		

Format Parameter	Not Used	"UnloadPart %1.0f TGID=%1.0f loc %1.0f TNOW=%5.2f " Entity.Type, IDENT, Entity.Station, TNOW
** **TASKS** Order Part Type 1 Format Parameter	 Not Used	 "OrderPartType %1.0f TNOW=%5.2f;" Entity.Type, TNOW
** **ARRIVALS** Order Part Type 1 Type TypeID Time Assignment Variable ID Value	Not Used	 Station Order Release Message Entity.Type Part Type 1

Keys

AGV# refers to the unit number of the AGV

* Similar settings and changes for Modules of 'Enter Cell 2', 'Enter Cell 3', 'Enter Cell 4' and 'Enter Exit Station'.

** Similar variable name and parameter settings for 'Order Part 2' and 'Order Part 3'

5 Conclusions

Once the key performance indicators (KPI) have been identified, converting a simulation model into an emulation model requires three fundamental changes. Firstly, the abstraction to the appropriate level of detail necessitates the model to be modularised and configurable. Secondly, Communication procedure in real time needs to be established. Finally before running the model, the simulation model needs to set up in real time execution mode.

The work presented here only shows the basic steps in converting the simulation model in Arena software environment. Nonetheless the methodology is also applicable in other simulation languages and packages.

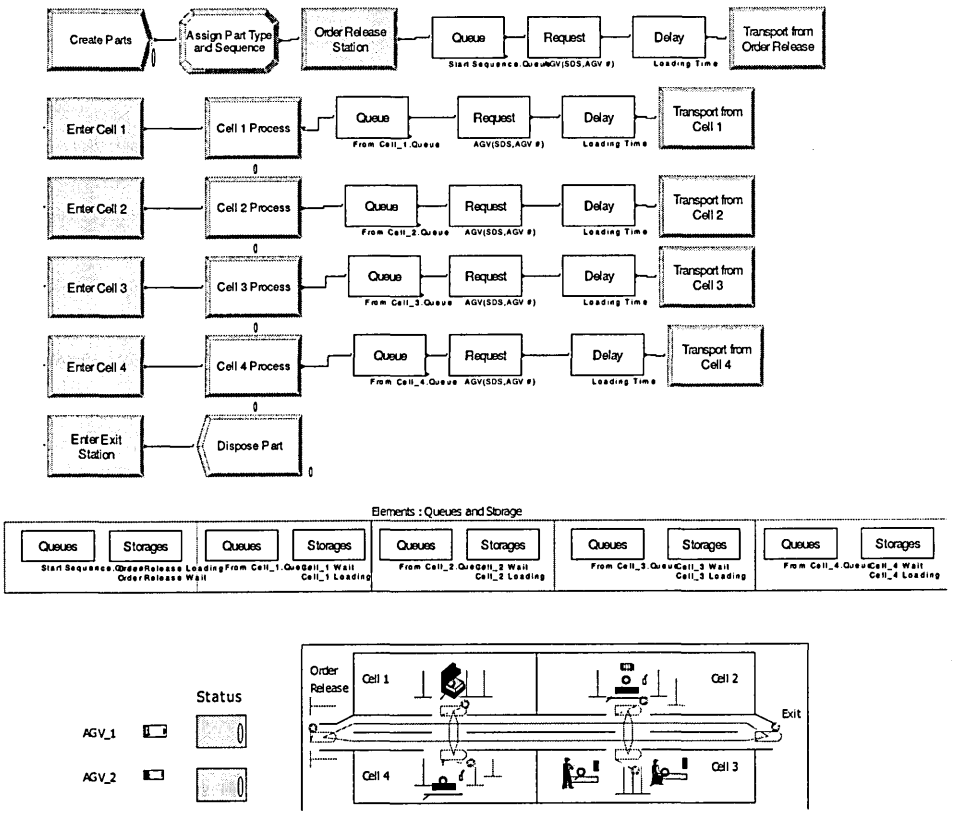


Fig.3 Simulation model

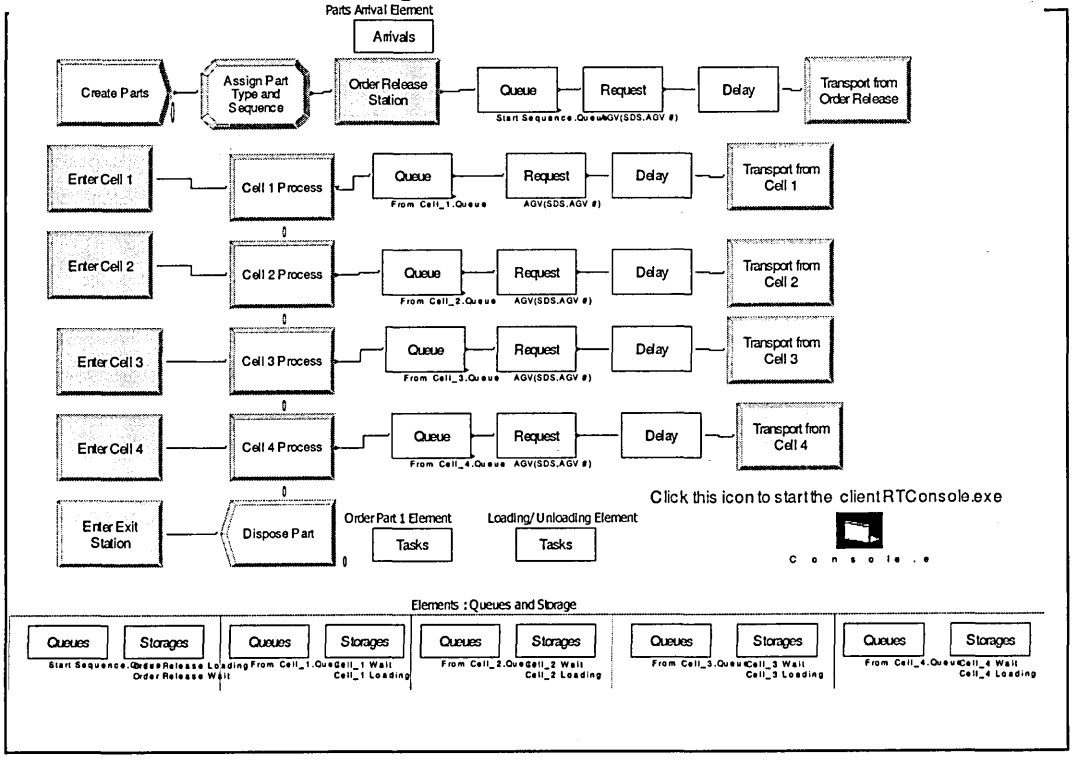


Fig. 4 Emulation model

6 References

1. Brennan, R.W. and B. Foroughi, *A Control Framework To Support Responsive Manufacturing.* International Journal of Agile Management Systems, 1999. **1**(3): p. 159-168.
2. Ramasesh, R., S. Kulkarni, and M. Jayakumar, *Agility In Manufacturing Systems: An Exploratory Modeling Framework And Simulation.* Journal of Integrated Manufacturing Systems, 2001. **12**(7): p. 534 - 548.
3. McGregor, I. and R.A. Walters. *Emulation Overview.* in *The 2001 Brooks Automation Symposium Proceedings.* 2001.
4. Mcgregor, I. *The Relationship Between Simulation And Emulation.* in *The 2002 Winter Simulation Conference.* 2002.
5. Bodner, D.A. and L.F. McGinnis. *A Structured Approach to Simulation Modeling of Manufacturing Systems.* in *The 2002 Industrial Engineering Research Conference.* 2002. Orlando.
6. Ball, P., *Abstracting Performance In hierarchical Manufacturing Simulation.* Journal Of Materials Processing Technology, 1998. **76**(1-3): p. 246-251.
7. Takus, D.A. and D.M. Profozich. *Arena Software Tutorial.* in *The 1997 Winter Simulation Conference.* 1997.

USER PERSPECTIVES ON THE USE OF EMULATION IN CONTROL SYSTEM TESTING

Khalid Hasnan, Terrence Perera, David Clegg
Faculty of Arts, Computing, Engineering and Sciences,
Sheffield Hallam University,
Sheffield S1 1WB, UK

ABSTRACT

Simulation models with high level of details also known as emulation has been used to test control in a variety of sectors such as manufacturing, logistics and transportation. To establish the nature of current use and investigate the factors that inhibit its use, a survey was conducted to obtain feedback from simulation practitioners on the use of emulation, their application areas and development problems. This paper presents findings of the survey and highlights potential research areas for the application of emulation as well as recommendations for development with regard to its model building.

1 INTRODUCTION

Traditionally, control systems are often only fully tested after commissioning, at the 'shop floor'. When this has to be done within the time constraints of the project's commissioning phase; the result is often unsatisfactory for all concerned, leading to project overrun, extended ramp times and rising costs. Thus, it is vital to test control systems before implementing them.

The benefits and potential of using emulation is very well acknowledged [1] Nonetheless there are also few concerns that need to be addressed. Among them the economy of scale of building such model and there is a need to understand on how to develop an emulation model in more cost effective manner.

The paper begins with a background discussion of using emulation for control system testing based on case studies. It is followed by the outline of the survey, the results and the analysis. The conclusions highlight the user's perspective on the current use of simulation for control system testing as well their expectations for the future.

2 SIMULATION FOR CONTROL SYSTEM TESTING

With the technology available today, a combination of reality and simulation to test control systems is seen to be appropriate. Four possible approaches to test control systems, based on the possible combinations between reality and simulation, are shown Figure 1. [2]

1. A combination of a control system and a system being controlled both in reality. The control system is tested during or after commissioning.

2. *Soft commissioning*. A combination of a control system in reality and a simulated system being controlled. A hardware-in-the-loop (HIL)-based approach where the inputs and outputs of a controller are connected to a simulation of the part to be controlled.
3. *Reality in the loop*. A combination of a simulated control system and a real system being controlled such as on a test bed.
4. *Off-line simulation*. A combination of both a simulated control system and a simulated system being controlled. Also referred to as pure simulation, it is often used to understand the behaviour of a system, or to predict an outcome under different internal and external influences.

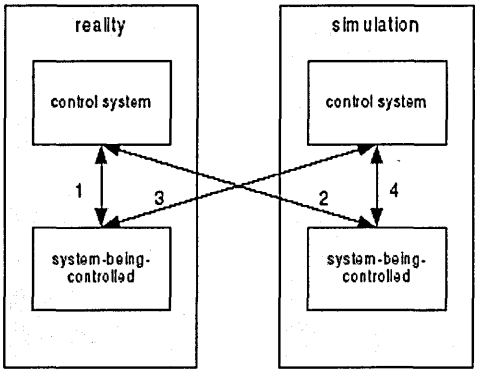


Fig. 1 Approaches for Testing Control Systems [3]

From initial case studies [3-6] , building a simulation model for the purpose of control system testing involves the following.

a) Determining the correct level of detail. Choosing the appropriate level of detail seems to be a balancing act between, minimising the details on the one hand and, adding details to ensure usefulness of the model on the other hand.
b) Modifying simulation code, to improve the quality of common core simulation functions, improve the potential for creating reusable modelling components from those core functions, and improve the integration of simulation packages with other applications including controllers.
c) Interfacing between modules. It involves the simulation software interfacing, synchronizing and real time capability.

3 SURVEY AMONG SIMULATION PRACTITIONERS

The main objectives of the survey were to investigate the extent of use of Emulation model for Control System Testing, its application areas and users opinion about its model building.

The questionnaire distributed to the participants of Winter Simulation Conference 2003 consisted of seven topics dealing with

8) type of user ,
9) whether or not using simulation for control system testing,
10) current application area,

11) ranking the benefits of using simulation for control system testing,
12) important stages for its model building,
13) simulation packages used
14) potential application areas.

Topics (3), (4) and (5) only apply to current users of simulation for control system testing. The aim was to gather information based on their experience the benefits of emulation model and facilities that would assist the development of emulation model.

The general meaning of each application area listed in the survey is as follows:
Process control involves monitoring, controlling and improving a process typically in production environment.[7, 8]
Business Process is collection of activities designed to produce a specific output for a particular customer or market. [9]
Material Handling is the movement, storage, control and protection of materials, goods and products throughout the process of manufacturing, distribution, consumption and disposal. [3, 10]
Security system is the mechanism to protect facilities against intrusions by external threats as well as unauthorized acts by insiders. It includes physical as well as information protection.[11, 12]
Transport system is the facility consisting of the roads and equipment necessary for the movement of passengers or goods. Mode of transport includes land, air and water.[13]

The questionnaire also contained additional space so that respondents could specify particular application areas, benefits and simulation packages that were not listed in the original choice of answers.

The survey sample was not selected by any formal statistical method. The respondents were participants of a simulation conference believed to be regular users of simulation. The ratio of responses from academics and industry was 40% to 60%.

4 RESULTS OF SURVEY

The results of the survey are presented in two sections. The first section presents results based on user's background, distinguishing academic and industrial users view. Second section focuses on presenting a general need and expectation on the use of emulation for control system testing.

4.1 Results based on user background

Application area of simulation for control system testing currently being used and possible application area in the future according to the user type are shown in Table 1

Table 1 Current and potential areas using of emulation for control system testing

		User Type			
		Academic (%)		Industrial (%)	
		Current	Potential	Current	Potential
Application area using emulation for control system testing	Process Control	28.6	27.6	33.3	33.3
	Business Process	28.6	17.2	14.3	18.2
	Material Handling	28.6	20.7	28.6	24.2
	Security System	0	10.3	0	3.0
	Transport System	14.3	24.1	23.8	21.2

4.2 Results based on users need and expectation

The distribution of the application area of emulation, present and future from simulation practitioner's perspective is shown in Table 2.

Table 2. Present and Future Application Area of Emulation

	Present (%)	Expected (%)
Process Control	33.3	30.9
Business Process	18.5	16.4
Material Handling	29.6	23.6
Security System	0	7.3
Transport System	18.5	21.8

The results of ranking the benefits of using emulation for control system testing based on weighted average calculation are as follows:

(1) Efficient use of resource
(2) Low overall cost
(3) Shorter commissioning time
(4) Client satisfaction

Ranking of the importance of development stage requiring specific tool for emulation model building would indicate (1) inhibiting factors at present, (2) important areas of research and development. The results of survey, in order of importance, are as follows:

(1) Interfacing between models,
(2) Modifying simulation code,
(3) Determining the correct level of detail.

5 ANALYSIS OF SURVEY RESULTS

The extent to which simulation is used for control system testing and the approach towards control system testing may vary between different types of users and

organizations, depending on the needs and practicality. This is indicated by the results shown in Table 1 and Table 2.

While process control and material handling are considered by both types of users to be the prime application areas of emulation, present and future their views are quite different as regards to other application areas.

On the future use as shown in Table 2, there is notable expectation in using emulation in the control system testing of transport system and security system among the academic users indicating research interest.

Another point to note is that the industrial users expects an increase in the use emulation in business process system testing. This can be viewed as increasing use of emulation as another tool towards increasing competitiveness among companies.

Regarding the main benefit of building emulation model, although it was initially perceived to be shortening the commissioning time [8, 10, 14], the results of the survey on users perception on the benefits of using emulation for control system testing indicated a different view. The ranking of putting efficient use of resource first, followed by low overall cost, shorter commissioning time and client satisfaction underlines the priority of concerns among the respondents.

Table 3 highlights the prospect of simulation model for control system testing across the spectrum of application areas, if there is better technology for its model building. It shows a shift towards a wider area of application covering areas beyond manufacturing and production. Thus, facilities for emulation model building for generic application is needed.

Regarding specific tools for emulation model building most respondents noted interfacing between models as most important. Works by researchers [2, 7, 15, 16] also highlights its importance. The ranking is followed by facility to modify simulation code and getting the correct level of detail. These facilities depended on the type and internal structure of simulation package being used. Nikoukaran [17] provides an insight towards this requirement.

Good system engineering practice would begin with a pure simulation and as components become better defined with the aid of simulation, they can be fabricated and replaced in the control loop. Once physical components are added to the loop, un-modelled characteristics can be investigated, and controls can be further refined.

Similar approach of control system testing could also be adapted to non engineering application as indicated by the response of the survey.

6 CONCLUSIONS

Emulation has the potential being used for control system testing in areas other than manufacturing and production like transport system, business process and security system based on a similar concept to Hardware in the loop simulation (HILS).

The simulation users acknowledged a need to provide a generic methodology and facilities for developing emulation models particularly regarding interfacing between models.

In general this survey has identified areas for development regarding using simulation for control system testing.

REFERENCES

1. Banks, J. *Simulation In The Future.* in *The 2000 Winter Simulation Conference.* 2000. Orlando, FL, USA.
2. Auinger, F., M. Vorderwinkler, and G. Buchtela. *Interface Driven Domain-Independent Modeling Architecture For "Soft-Commissioning" And "Reality In The Loop".* in *The 1999 Winter Simulation Conference.* 1999. Phoenix, AZ, USA.
3. Versteegt, C. and A. Verbraeck. *The Extended Use Of Simulation In Evaluating Real-Time Control Systems Of AGVS And Automated Material Handling Systems.* in *The 2002 Winter Simulation Conference.* 2002. Salt Lake City, UT, USA.
4. Mcgregor, I. *The Relationship Between Simulation And Emulation.* in *The 2002 Winter Simulation Conference.* 2002. Salt Lake City, UT, USA.
5. Persson, J.F., *The impact of different levels of detail in manufacturing systems simulation models.* Journal of Robotics and Computer Integrated Manufacturing, 2002. **18**: p. 319-325.
6. Ball, P., *Abstracting Performance In hierarchical Manufacturing Simulation.* Journal Of Materials Processing Technology, 1998. **76**(1-3): p. 246-251.
7. Davis, W.J., J.G. Macro, A.L. Brook, M.S. Lee, and G.S. Zhou. *Developing A Real-Time Emulation/Simulation Capability For The Control Architecture To The Ramp FMS.* in *The 1996 Winter Simulation Conference.* 1996. Coronado, CA, USA.
8. LeBaron, H.T. and R.A. Hendrickson. *Using Emulation To Validate A Cluster Tool Simulation Model.* in *The 2001 Winter Simulation Conference.* 2001. Arlington, VA, USA.
9. Aguilar-Savén, R.S.R.S., *Business process modelling: Review and framework.* International Journal of Production Economics, 2004. **90**(2): p. 129-149.
10. Mueller, G. *Using Emulation To Reduce Commissioning Costs On A High Speed Bottling Line.* in *The 2001 Winter Simulation Conference.* 2001. Arlington, VA, USA.
11. Jordan, S.E., M.K. Snell, M.M. Madsen, J.S. Smith, and B.A. Peters. *Discrete-Event Simulation For The Design And Evaluation Of Physical Protection Systems.* in *The 1998 Winter Simulation Conference.* 1998. Washington DC, USA.
12. Smith, J.S., B.A. Peters, S.E. Jordan, and M.K. Snell. *Distributed Real-Time Simulation For Intruder Detection System Analysis.* in *The 1999 Winter Simulation Conference.* 1999. Phoenix, AZ, USA.

13. Verbraeck, A. and C. Versteegt. *Logistic for fully automated large-scale freight transport systems.* in *2001 IEEE Intelligent Transportation Systems Conference.* 2001. Oakland (CA), USA.
14. Schiess, C. *Emulation : Debug in the Lab - Not On The Floor.* in *The 2001 Winter Simulation Conference.* 2001. Arlington, VA, USA.
15. Rogers, P. and R.W. Brennan. *A Simulation Testbed For Comparing The Performance Of Alternative Control Architectures.* in *The 1997 Winter Simulation Conference.* 1997. Atlanta, GA, USA.
16. Schludermann, H., T. Kirchmair, and M. Vorderwinkler. *Soft-Commissioning: Hardware-In-The-Loop-Based Verification Of Controller Software.* in *The 2000 Winter Simulation Conference.* 2000. Orlando, FL, USA.
17. Nikoukaran, J., V. Hlupic, and R.J. Paul, *A Hierarchical Framework For Evaluating Simulation Software.* Simulation Practice and Theory, 1999. 7: p. 219 - 231.

Appendix C

Arena Model ThisDocument and FrmConnect For Emulation Model MfgPlant_AGV_RT.doe

Appendix C.1 **Arena ThisDocument Object VBA Code**

Appendix C.2 **VBA Code for userform 'frmConnect'**

Appendix C.1 Arena ThisDocument Object VBA Code

```
Option Explicit

'Global declarations
Public blnMessageWaiting As Boolean 'Is there a message from the client that needs to be
processed?
Public blnIsConnected As Boolean 'Is the window socket connected to the remote host?
Public strMessage As String 'Most recent message from the client

Private Function ModelLogic_RealTimeInitialize(ByVal processName As String, ByVal
remoteProcessName As String) As Long
'
'RealTimeInitialize
'
'If Arena is running in execution mode, then this event is
'automatically called once by Arena before the first replication.
'Place code that initializes the communication port with the external client in
'this function.
'
With frmConnect
'Display frmConnect to prompt for the IP address and port.
   .TextBox1.Text = "127.0.0.1"
   .TextBox2.Text = "8111"
   .Show
End With

End Function

Private Function ModelLogic_RealTimeReceive(message As String) As Long
'
'RealTimeReceive
'
'If Arena is running in execution mode, then this event is
'periodically called by Arena (e.g., every .1 seconds).
'Place code that checks for incoming messages from the client
'in this function.
'
If (blnMessageWaiting = True) Then
'If a message has been received from the client, then store the message in the parameter
'"message" so that it may be processed by Arena.
  blnMessageWaiting = False
  message = strMessage
End If
End Function

Private Function ModelLogic_RealTimeSend(ByVal message As String) As Long
'
```

```
'RealTimeSend
'
'If Arena is running in execution mode, then this event is
'automatically called by Arena each time a message (stored in the argument "message")
'is sent by an entity from a DELAY, ROUTE, TRANSPORT, or MOVE block.
'Place code that sends messages to the client in this function.
'
'The format of the argument "message" is defined in the TASKS element.
'
If (Not blnIsConnected) Then
'End the model run. The socket is not connected.
   Arena.ActiveModel.End
Else
'Send the message string over the window socket to the client
   frmConnect.Winsock1.SendData (message + ";")

End If
End Function

Private Function ModelLogic_RealTimeTerminate() As Long
'
'RealTimeTerminate
'
'If Arena is running in execution mode, then this event is automatically called
'once by Arena at the end of the last replication. Place code that terminates
'communications with the client in this function.
'
'Close socket if not already closed
If (frmConnect.Winsock1.State <> sckClosed) Then
  frmConnect.Winsock1.Close
End If

End Function
```

Appendix C.2 VBA Code for userform 'frmConnect'

```
Option Explicit

Private Sub CommandButton1_Click()
'
'"Connect" button click event
'
On Error Resume Next
'Close socket if not already closed
If (Winsock1.State <> sckClosed) Then
  Winsock1.Close
End If
'Call the Connect method on Winsock1 (winsock object) to connect to the remote host.
Winsock1.Connect TextBox1.Text, TextBox2.Text

'NOTE TO USER: If the compile error "Variable not defined" is occuring on the object
'`Winsock1` in this subroutine, then the Microsoft winsock control `Winsock1` is
'missing from the form `frmConnect`. To run using the VBA
'code, a Microsoft winsock control named `Winsock1` must first be added to
'the form `frmConnect`. Double-Click `frmConnect` in the project explorer and
'select View/Toolbox to add `Winsock1`.
'
'Click the icon "Instructions" in the model window for step-by-step instructions on
'running this example.
End Sub

Private Sub Winsock1_Connect()
'This winsock event is called after successful connection to remote host
'
MsgBox "Successfully connected to RTConsole.", vbInformation
ThisDocument.blnIsConnected = True
'Hide this form
frmConnect.Hide
End Sub

Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
' This winsock event is called automatically when data arrives to the socket.
'
Dim str As String

'Store message in str variable.
frmConnect.Winsock1.GetData str, vbString
'Set the flag blnMessageWaiting to TRUE so that Arena will know a message needs
```

```
'to be processed the next time the RealTimeReceive event is called.
ThisDocument.blnMessageWaiting = True
'Store message in global strMessage so Arena can get string later
ThisDocument.strMessage = str
End Sub

Private Sub Winsock1_Error(ByVal Number As Integer, Description As String, ByVal Scode
As Long, ByVal Source As String, ByVal HelpFile As String, ByVal HelpContext As Long,
CancelDisplay As Boolean)
'This winsock event is called after an error
'
'Display error message
MsgBox Description & Chr(10) & Chr(10) & _
"Error connecting to RTConsole.exe. Before running this model in execution mode," &
Chr(10) & _
" First start the client RTConsole.exe and set it to listen. Click the icon Instructions for" &
Chr(10) & _
"step-by-step instructions on running this example.", vbCritical, "RT Execution Mode.doe"
'Hide this form
frmConnect.Hide
End Sub
```

Appendix D

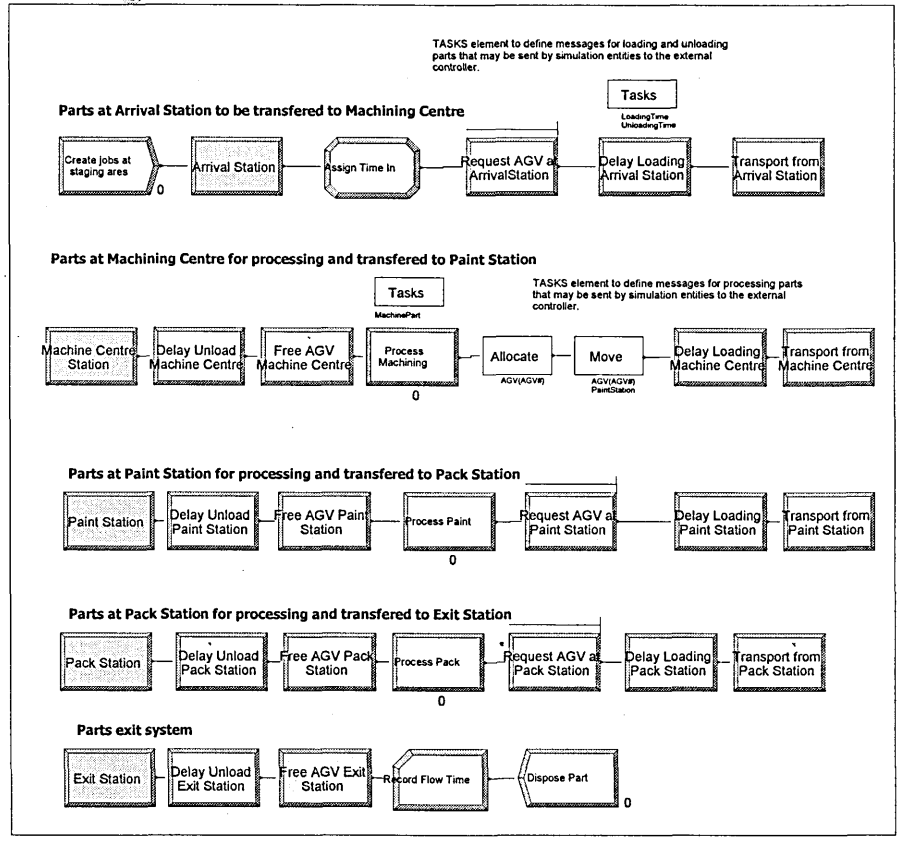
Emulation Model

Appendix D.1 Logic Flowchart and Animation

Appendix D.2 SIMAN code

Appendix D.1 Logic Flowchart and Animation

Logic



MfgPlant_AGV_RT.doe

Description and Instructions

Manufacturing Plant Emulation

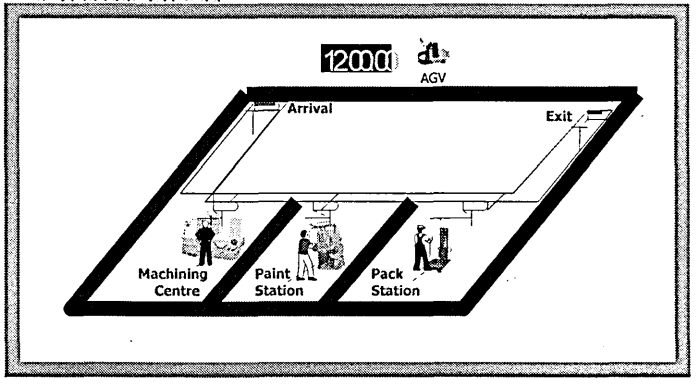
Click this icon to view step-by-step instructions for running this model

Instructions.txt

Click this icon to start RTConsole.exe

RTConsole.exe

Animation



Appendix D.2 SIMAN code

```
;
;
;   Model statements for module:  Station 6
;
2$          STATION,     ExitStation;
37$         DELAY:       0.0,,VA:NEXT(3$);
;
;
;   Model statements for module:  Delay 9
;
3$          DELAY:       Exit_UnloadingTime,,Other:NEXT(4$);
;
;
;   Model statements for module:  Free 5
;
4$          FREE:        AGV(AGV#):NEXT(1$);
;
;
;   Model statements for module:  Record 2
;
1$          TALLY:       FlowTime,INT(TimeIn),1:NEXT(0$);
;
;
;   Model statements for module:  Dispose 2
;
0$          ASSIGN:      Dispose Part.NumberOut=Dispose Part.NumberOut + 1;
38$         DISPOSE:     Yes;
;
;
;   Model statements for module:  Station 7
;
10$         STATION,     PackStation;
41$         DELAY:       0.0,,VA:NEXT(11$);
;
;
;   Model statements for module:  Delay 11
;
11$         DELAY:       Pack_UnloadingTime,,Other:NEXT(12$);
;
;
;   Model statements for module:  Free 6
;
12$         FREE:        AGV(AGV#):NEXT(5$);
```

```
;
;
;   Model statements for module:  Process 4
;
5$          ASSIGN:       Process Pack.NumberIn=Process Pack.NumberIn + 1:
                          Process Pack.WIP=Process Pack.WIP+1;
71$         STACK,        1:Save:NEXT(45$);

45$         QUEUE,        Process Pack.Queue;
44$         SEIZE,        2,VA:
                          Packer,1:NEXT(43$);

43$         DELAY:        Triangular(1.8,2.2,2.6),,VA:NEXT(86$);

86$         ASSIGN:       Process Pack.WaitTime=Process Pack.WaitTime + Diff.WaitTime;
50$         TALLY:        Process Pack.WaitTimePerEntity,Diff.WaitTime,1;
52$         TALLY:        Process Pack.TotalTimePerEntity,Diff.StartTime,1;
76$         ASSIGN:       Process Pack.VATime=Process Pack.VATime + Diff.VATime;
77$         TALLY:        Process Pack.VATimePerEntity,Diff.VATime,1;
42$         RELEASE:      Packer,1;
91$         STACK,        1:Destroy:NEXT(90$);

90$         ASSIGN:       Process Pack.NumberOut=Process Pack.NumberOut + 1:
                          Process Pack.WIP=Process Pack.WIP-1:NEXT(6$);
;
;
;   Model statements for module:  Request 5
;
6$          QUEUE,        Request AGV at Pack Station.Queue;
            REQUEST,      1:AGV(CYC,AGV#),50:NEXT(8$);
;
;
;   Model statements for module:  Delay 10
;
8$          DELAY:        Pack_LoadingTime,,Other:NEXT(9$);
;
;
;   Model statements for module:  Transport 5
;
9$          TRANSPORT:    AGV,ExitStation,50;
;
;
;   Model statements for module:  Station 8
;
15$         STATION,      PaintStation;
96$         DELAY:        0.0,,VA:NEXT(16$);
;
;
;   Model statements for module:  Delay 13
;
16$         DELAY:        Paint_UnloadingTime,,Other:NEXT(17$);
```

```
;
;
;   Model statements for module:  Free 7
;
17$         FREE:         AGV(AGV#):NEXT(34$);
;
;
;   Model statements for module:  Process 7
;
34$         ASSIGN:       Process Paint.NumberIn=Process Paint.NumberIn + 1:
                          Process Paint.WIP=Process Paint.WIP+1;
126$        STACK,        1:Save:NEXT(100$);

100$        QUEUE,        Process Paint.Queue;
99$         SEIZE,        2,VA:
                          Painter,1:NEXT(98$);

98$         DELAY:        Triangular(1.8,2.2,2.6),,VA:NEXT(141$);

141$        ASSIGN:       Process Paint.WaitTime=Process Paint.WaitTime + Diff.WaitTime;
105$        TALLY:        Process Paint.WaitTimePerEntity,Diff.WaitTime,1;
107$        TALLY:        Process Paint.TotalTimePerEntity,Diff.StartTime,1;
131$        ASSIGN:       Process Paint.VATime=Process Paint.VATime + Diff.VATime;
132$        TALLY:        Process Paint.VATimePerEntity,Diff.VATime,1;
97$         RELEASE:      Painter,1;
146$        STACK,        1:Destroy:NEXT(145$);

145$        ASSIGN:       Process Paint.NumberOut=Process Paint.NumberOut + 1:
                          Process Paint.WIP=Process Paint.WIP-1:NEXT(32$);
;
;
;   Model statements for module:  Request 9
;
32$         QUEUE,        Request AGV at Paint Station.Queue;
          REQUEST,        1:AGV(CYC,AGV#),50:NEXT(13$);
;
;
;   Model statements for module:  Delay 12
;
13$         DELAY:        Paint_LoadingTime,,Other:NEXT(14$);
;
;
;   Model statements for module:  Transport 6
;
14$         TRANSPORT:    AGV,PackStation,50;
;
;
;   Model statements for module:  Station 9
;

19$         STATION,      MachineCentre;
151$        DELAY:        0.0,,VA:NEXT(20$);
```

```
;
;
;   Model statements for module:  Delay 14
;
20$         DELAY:      TASKID(EXPO( 3 ),UnloadingTime),,Other:NEXT(21$);
;
;
;   Model statements for module:  Free 8
;
21$         FREE:       AGV(AGV#):NEXT(18$);
;
;
;   Model statements for module:  Process 6
;
18$         ASSIGN:     Process Machining.NumberIn=Process Machining.NumberIn + 1:
                        Process Machining.WIP=Process Machining.WIP+1;
181$        STACK,      1:Save:NEXT(155$);

155$        QUEUE,      Process Machining.Queue;
154$        SEIZE,      2,VA:
                        Machinist,1:NEXT(153$);

153$        DELAY:      TASKID(UNIF(.5,1.5),MachinePart),,VA:NEXT(196$);

196$        ASSIGN:     Process Machining.WaitTime=Process Machining.WaitTime +
Diff.WaitTime;
160$        TALLY:      Process Machining.WaitTimePerEntity,Diff.WaitTime,1;
162$        TALLY:      Process Machining.TotalTimePerEntity,Diff.StartTime,1;
186$        ASSIGN:     Process Machining.VATime=Process Machining.VATime +
Diff.VATime;
187$        TALLY:      Process Machining.VATimePerEntity,Diff.VATime,1;
152$        RELEASE:    Machinist,1;
201$        STACK,      1:Destroy:NEXT(200$);

200$        ASSIGN:     Process Machining.NumberOut=Process Machining.NumberOut +
1:
                        Process Machining.WIP=Process Machining.WIP-1:NEXT(30$);

30$         ALLOCATE,   1:AGV(AGV#),MachineCentre;
31$         MOVE:       AGV(AGV#),PaintStation:NEXT(22$);


;
;
;   Model statements for module:  Delay 15
;
22$         DELAY:      TASKID(EXPO( 3),LoadingTime),,Other:NEXT(23$);
;
;
;   Model statements for module:  Transport 7
;
23$         TRANSPORT:  AGV,PaintStation,50;
```

```
;
;
;   Model statements for module:  Create 2
;

203$        CREATE,       1,MinutesToBaseTime(0.0),Entity
1:MinutesToBaseTime(EXPO(3)):NEXT(204$);

204$        ASSIGN:       Create jobs at staging ares.NumberOut=Create jobs at staging
ares.NumberOut + 1:NEXT(24$);
;
;
;   Model statements for module:  Station 10
;

24$         STATION,      ArrivalStation;
209$        DELAY:        0.0,,VA:NEXT(25$);
;
;
;   Model statements for module:  Assign 2
;
25$         ASSIGN:       TimeIn=TNOW:NEXT(26$);
;
;
;   Model statements for module:  Request 8
;
26$         QUEUE,        Request AGV at ArrivalStation.Queue;
          REQUEST,        1:AGV(CYC,AGV#),50:NEXT(28$);
;
;
;   Model statements for module:  Delay 16
;
28$         DELAY:        TASKID(EXPO( 3),LoadingTime),,Other:NEXT(29$);
;
;
;   Model statements for module:  Transport 8
;
29$         TRANSPORT:    AGV,MachineCentre,50;
```

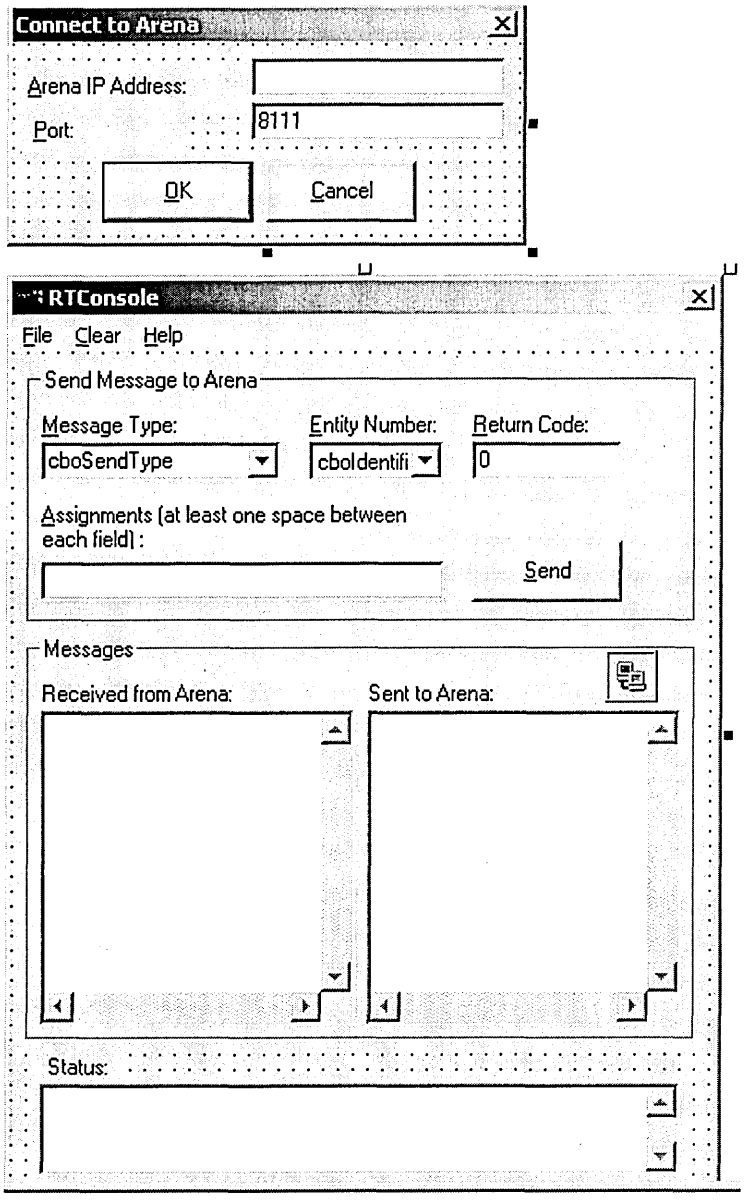
Appendix E.1 RTConsoleMain.frm

```
Option Explicit
Private Sub cboMessageType_LostFocus()
Call cboMessageType_Click
End Sub
Private Sub cmdSend_Click()
'Send a Message to Arena

'Declarations
Dim blnResponseToTask As Boolean
Dim blnEntityIdentifier As Boolean
Dim strMessage As String

blnResponseToTask = (cboMessageType.Text = "0") Or (cboMessageType.Text = "0 - Response to
Task")
blnEntityIdentifier = cboIdentifier.Text <> ""
'If responding to a task, must specify entity identifier
If (blnResponseToTask And Not blnEntityIdentifier) Then Exit Sub

'Store message to send in strMessage
If blnResponseToTask Then
   strMessage = "0 " & cboIdentifier.Text & " " & txtReturnCode.Text
   'Remove entity identifier from combobox
   cboIdentifier.RemoveItem (cboIdentifier.ListIndex)
Else
   strMessage = cboMessageType.Text & " " & txtAssignments.Text
End If
'Send the message
wsArena.SendData (strMessage + Chr(0))
AddLineToTextBox txtSent, strMessage

End Sub
Private Sub cboMessageType_Click()

'Declarations
Dim blnResponseToTask As Boolean

'Response to task?
blnResponseToTask = (cboMessageType.Text = "0") Or (cboMessageType.Text = "0 - Response to
Task")

'Set visible properties of Send Message fields
txtAssignments.Visible = Not blnResponseToTask
lblAssignments.Visible = Not blnResponseToTask
cboIdentifier.Visible = blnResponseToTask
txtReturnCode.Visible = blnResponseToTask
lblIdentifier.Visible = blnResponseToTask
lblReturnCode.Visible = blnResponseToTask
End Sub
Private Sub Form_Load()
```

```
'Setup MessageType combo-box
cboMessageType.AddItem "0 - Response to Task"
cboMessageType.AddItem "1"
cboMessageType.AddItem "2"
cboMessageType.AddItem "3"
cboMessageType.AddItem "4"
cboMessageType.AddItem "5"
cboMessageType.Text = "0 - Response to Task"
Call cboMessageType_Click
End Sub
Private Sub Form_Terminate()
'Close socket if not already closed
If (frmMain.wsArena.State <> sckClosed) Then
  frmMain.wsArena.Close
End If
End Sub
Private Sub menuClearAll_Click()
'Clear all textboxes
txtSent.Text = ""
txtReceived.Text = ""
txtStatus.Text = ""
End Sub
Private Sub menuClearReceived_Click()
'Clear Messages Received textbox
txtReceived.Text = ""
End Sub
Private Sub menuClearSent_Click()
'Clear Messages Sent textbox
txtSent.Text = ""
End Sub
Private Sub menuClearStatus_Click()
'Clear Status textbox
txtStatus.Text = ""
End Sub
Private Sub menuFileDisconnect_Click()
'Close socket if not already closed
If (frmMain.wsArena.State <> sckClosed) Then
  frmMain.wsArena.Close
End If
'Enable Listen and Connect menu items
MenuFileListen.Enabled = True
MenuFileConnect.Enabled = True
'Disable Send button and Disconnect Menu Item
MenuFileDisconnect.Enabled = False
cmdSend.Enabled = False
'Clear EntityIdentifier combo-box
cboIdentifier.Clear

AddLineToTextBox txtStatus, "Socket disconnected"

End Sub
Private Sub menuFileListen_Click()
'Set Socket to listen on port

'Declarations
Dim strPort As String
Dim intRet As Integer
```

```
On Error Resume Next
'Prompt for port number
strPort = InputBox("", "Listen on Port:", "8111")

If (Len(strPort) > 0) Then
   wsArena.LocalPort = CLng(strPort)
   'Check for error assigning LocalPort property
   If (Err.Number <> 0) Then
      intRet = MsgBox("Error assigning port. Port """ & strPort & """ may be invalid.", vbCritical)
      GoTo ExitHere
   End If
   'Set socket to listen
   wsArena.Listen

   AddLineToTextBox txtStatus, "Socket listening on port " & strPort

   'Disable Listen and Connect menu items
   MenuFileListen.Enabled = False
   MenuFileConnect.Enabled = False
   'Enable Disconnect menu item
   MenuFileDisconnect.Enabled = True
End If

ExitHere:
   On Error GoTo 0
End Sub
Private Sub MenuFileConnect_Click()
'Connect to Arena
frmConnect.Show vbModal
End Sub
Private Sub menuFileExit_Click()
Unload Me
End Sub
Private Sub menuHelpAbout_Click()
frmAbout.Show
End Sub
Private Sub wsArena_Connect()
'Winsock event called after successful connection to remote host

'Enable Disconnect menu item and Send button
MenuFileDisconnect.Enabled = True
cmdSend.Enabled = True
'Disable Connect and Listen menu items
MenuFileConnect.Enabled = False
MenuFileListen.Enabled = False

AddLineToTextBox txtStatus, "Socket connected to port " & wsArena.RemotePort & " of host " _
& wsArena.RemoteHostIP
End Sub
Private Sub wsArena_ConnectionRequest(ByVal requestID As Long)
'Winsock event called when a remote client is attempting to connect

If wsArena.State <> sckClosed Then wsArena.Close
wsArena.Accept requestID

'Enable Send button
```

```
cmdSend.Enabled = True

AddLineToTextBox txtStatus, "Socket accepted connection request on port " & wsArena.LocalPort
End Sub
Private Sub wsArena_DataArrival(ByVal bytesTotal As Long)
'Winsock event called when data is received from remote computer

'Declarations
Dim strData As String
Dim strLines() As String
Dim strTokens() As String
Dim lngI As Long

'Store socket data in strData
wsArena.GetData strData, vbString
'First remove any line feed characters from string
strLines = Split(strData, Chr(10))
strData = Join(strLines)
'Now parse data into the individual lines (Each line should end with a ";" character)
strLines = Split(strData, ";")

'Process each line
For lngI = 0 To UBound(strLines)
   'Remove ending and leading spaces (if any)
   strLines(lngI) = Trim(strLines(lngI))
   If (Len(strLines(lngI)) > 0) Then
      'Add line to Messages Received textbox
      AddLineToTextBox txtReceived, strLines(lngI)
      'Parse line into tokens (Each token should be separated by a space)
      strTokens() = Split(strLines(lngI), " ")
      'Filter string array to only the "TGID" token
      strTokens = Filter(strTokens, "TGID=")
      If (UBound(strTokens) > -1) Then
      'There is a "TGID" token. Add the entity identifier
      'to the combo-box "Entity Identifier" (so that the user can select
      'this value when sending a response back to Arena
         cboIdentifier.AddItem Right(strTokens(0), Len(strTokens(0)) - 5)
      End If
   End If
Next

End Sub
Private Sub wsArena_Error(ByVal Number As Integer, Description As String, ByVal Scode As Long,
ByVal Source As String, ByVal HelpFile As String, ByVal HelpContext As Long, CancelDisplay As
Boolean)
'Winsock event called after an error
MsgBox ("Error " & Number & Chr(13) & Description)
'Close the socket
wsArena.Close
End Sub
Private Sub AddLineToTextBox(txtBox As TextBox, strLine As String)
'Adds strLine to txtBox
txtBox.Text = strLine + Chr(13) + Chr(10) + txtBox.Text
End Sub
```


Connect to Arena
Arena IP Address:
Port:
8111
OK
Cancel
RTConsole
File Clear Help
Send Message to Arena
Message Type:
cboSendType
Entity Number:
cboIdentifi
Return Code:
0
Assignments (at least one space between each field) :
Send
Messages
Received from Arena:
Sent to Arena:
Status:

Appendix E.2 RTConsoleConnect.frm

```
Option Explicit
Private Sub cmdCancel_Click()
Unload Me
End Sub
Private Sub cmdOK_Click()

On Error Resume Next
'Close socket if not already closed
If (frmMain.wsArena.State <> sckClosed) Then
  frmMain.wsArena.Close
End If

'Connect to remote host
frmMain.wsArena.Connect txtRemoteHost.Text, txtRemotePort
'Check for error
If (Err.Number <> 0) Then
   MsgBox ("Invalid IP address or port.")
   GoTo ExitHere
End If
Unload Me

ExitHere:
   On Error GoTo 0
End Sub
Private Sub Form_Load()
'Set default IP Address and Port
txtRemoteHost.Text = frmMain.wsArena.LocalIP
txtRemotePort.Text = "4334"
End Sub
```

Appendix F

Instructions for Running MfgPlant_RT_AGV.doe

ON-LINE HELP NOTE: Refer to the help topic "Arena RT" in the on-line help for a detailed description of Arena RT and its features.

There are two alternative approaches for implementing the inter-process communications
between Arena and an external client: through VBA or through user-code.
This example demonstrates VBA approach.

****
Instructions for running MfgPlant_RT_AGV.doe using VBA realtime events:
****

1. Enter the Visual Basic Editor and double-click the form frmConnect in the project explorer.

2. Make sure a Microsoft winsock control named `Winsock1` has been added to the form frmConnect. If the winsock control has not been added, select View/Toolbox to access the controls tab. Then drag-and-drop a winsock control onto the form. By default, it should be named `Winsock1`.

   Note that you must have a design time license of the Microsoft winsock control to add it to the project.

3. Spend some time browsing through the VBA code. Then exit the Visual Basic Editor.

4. In <Run Setup\Run Control>, verify that the model is set to "Run in Execution Mode".

5. In <Run\Setup\Run Control>, verify that "Load User-Coded .DLL" is NOT checked.

6. In <Run Setup\Run Speed>, verify that the model will "Advance Simulation Time Using a Real Time Factor" of 1.

7. Let's now run the example. First, click the icon "RTConsole.exe" in the named view "Model Description and Instructions" of the model. This starts the client application that will send and receive messages from Arena.

8. From RTConsole's menu bar, click <File\Listen...>. Keep the default of "8111" for the port and hit <OK>. RTConsole is now waiting to connect to Arena.

9. From Arena's menu bar, click <Run\Go> and begin running MfgPlant_RT_AGV.doe. Note that the VBA event RealTimeInitialize is called first, displaying a dialog asking for RTConsole's IP address and port. Keep the defaults and press <Connect>. The model should display a "Successfully connected to RTConsole" message that indicates a successful connection. Press <OK>.

10. The model is now running. Notice that the clock is advancing in real-time.

11. Adjust the application windows of Arena and RTConsole such that both of them are displayed on the screen. If you haven't already, press <A> in Arena to zoom to the model animation.

12. Notice that entities representing parts have entered the system at time 0.0 and sent "LoadPart" messages to RTConsole. Once 'loaded' the part moves with the mover to the designated destination. When it reaches the next station it sends a message "UnloadPart" parts begin processing on the first machines in their sequence, they also send "ProcessPart" messages to RTConsole.

13. A response is required from RTConsole for each of the "LoadPart " messages. A response indicates the entity has completed its loading in the "real system", and can now proceed to the next step in its sequence.

14. To respond to a "LoadPart " message sent by an entity on a mover, in RTConsole select the entity you want to respond to from the Entity Number pull-down list. Press <Send> to send the response.

15. Watch the entity you responded to in the model animation. It is now proceeding to the next step in its sequence which is travelling in the transporter to the designated station. In the Messages Sent to Arena list box of RTConsole, note the actual format of the response message sent back to Arena (e.g., "0 2 0" or "0 6 0"). A more detailed description of the format of messages sent to Arena may be found in the online help.

16. Through RTConsole, continue entering parts into the simulation and responding to "UnloadPart" , "MachinePart" messages etc. The simulation is emulating the "real system"!

17. At the end of the session, stop the model.

18. From RTConsole's menu bar, click <File\Disconnect> to close the socket.