REFERENCE

Sheffield Hallam University

Modular Development of Manufacturing Simulation Models

By
Alan Che Kien Kwan

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

July 2002

Abstract

Modular Development of Manufacturing Simulation Models

It is common practice within manufacturing companies to create simulation models at different time periods. These models are often used to represent various parts of the manufacturing systems. In general, these pre-built simulation models are required to be integrated together in order to evaluate the entire manufacturing system, this is not a simple task. This research addresses the issues involved in the integration of pre-built simulation models.

An in depth literature review was carried out to identify current strategies to overcome these issues. Based on structured research work, a set of recommendations is proposed to ensure easy integration of models. This set of recommendations will help simulation practitioners to minimise the errors occurred during the integration of simulation models.

The findings conclude more effort is required than is anticipated by most model builders and involves far more than 'just simply changing' the name of variables. A set of recommendations is therefore proposed to cope with the complexity and understanding of manufacturing systems. The research focuses on manufacturing systems but in general can be applied elsewhere.

Acknowledgements

This MPhil research was completed as part of the Master of Philosophy at the School of Engineering, Sheffield Hallam University, between October 1999 and July 2002. The work is my own and the results obtained during the MPhil program are to the best of my knowledge, original, where the work of others is used or drawn on and it is attributed to the relevant source.

I would like personally to express all my sincere gratitude to the people concerned with this research that was carried out. In particular my project supervisor Professor Terrence Perera for his input, guidance and patience throughout this programme. I would also like to thank my family and friends for their support, especially Mr Kenneth O'Riorden and Ms Susan Tran, who has given their support and patience throughout this programme.

A.C.K. Kwan
July 2002

Contents

	Page
Title Page	i
Abstract	ii
Acknowledgements	iii
Contents	iv
1. Introduction	
1.1. Introduction	1
1.2. Simulation modelling of large and complex manufacturing systems	1
1.3. Modular development of large scale manufacturing simulation models	2
1.4. Potential issues with modularity	2
1.5. The need for research	3
1.6. Contribution to the practice of simulation	4
1.7. Objectives of the project	
1.7.1. Identify the issues involved in the simulation of large-scale manufacturing systems.	4
1.7.2. Construct a series of modular simulation models.	4
1.7.3. Study of the issues involved with the integration of the modules developed in section 1.7.2.	5
1.7.4. Develop a framework to enable the modular development of manufacturing simulation models.	5
1.8. Delimitation's of scope and key assumptions	6
2. Literature Review	
2.1. Definition of simulation	7
2.2. Different types of simulation models	8
2.2.1. Static and dynamic	8
2.2.2. Deterministic and stochastic	8
2.2.3. Continuous and discrete	8
2.3. Discrete event simulation	9
2.4. Trends of simulation	10
2.5. Industrial Benefits	12
2.5.1. Applications	12

2.6. Issues in simulation 13
2.6.1. Issues: Limits in modelling capacity 13
2.6.2. Issues: Validation and verification 14
2.6.3. Issues: Incorrect data 15
2.6.4. Issues: Inappropriate use of animation 15
2.6.5. Issues: Modularity 16
2.7. Modularity and the modelling process 16
2.7.1. Model formulation 17
2.7.2. System investigation 17
2.7.3. Model representation 17
2.7.4. Model programming 17
2.7.5. Verification and Validation 17
2.7.6. Experiments 18
2.7.7. Documentation 18
2.8. Complex systems 18
2.9. Modular simulation 19
2.10. Potential benefits of modularization for manufacturing industries 20
2.11. Work done in modular simulation 22
2.11.1. Object oriented simulation 23
2.11.2. Distributed simulation 29
2.12. Conclusion 31

3. Methodology

3.1. Introduction 32
3.2. Selection of software 32
3.3. Arena simulation package 34
3.4. Conclusion 36

4. Experimentation

4.1. Introduction 37
4.2. Phase 1: Analysis of Arena building commands 37
4.3. Phase 2: Elimination of Logic building blocks 49
4.4. Phase 3: Experimentation 49
4.5. Phase 4: Recommendations 50

4.6. Phase 5: Conclusion 50
4.7. An example of experiments conducted 50

5. Results

5.1. Introduction 54
5.2. Summary of results 54
5.2.1. Common template results 54
5.2.2. Support template results 55
5.2.3. Transfer template results 56
5.2.4. Element template results 56
5.3 Integration issues 57
5.3.1. Integration issues 57
5.3.2. Further detail 63
5.3.3. Classification of issues 67
5.3.4. Conclusion 68

6. Discussion

6.1. Introduction 71
6.2. Experimental Issues 71
6.2.1. Animation 71
6.2.2. Mark time attribute 72
6.2.3. Unique numbers 72
6.2.4. Menu building block 72
6.2.5. Network links (animation) 72
6.2.6. Parameters within CREATE building block 73
6.3. Other issues 73
6.3.1. Visual Basic Application (VBA) 73
6.3.2. Documentation 77
6.3.3. Fundamentals of encapsulation 77
6.3.4. Redefinition of input variables 78
6.4. Discussion of hypothesis 78
6.5. Implications for industry 79
6.5.1. Customers and users 79
6.5.2. Modellers and analysts 79
6.5.3. Software vendors 80

6.6. Limitations 80
6.7. Recommendations 80
6.7.1. Recommendations for Type I & II 82
6.7.2. Recommendations for Type III 83
6.7.3. Recommendations for other issues 84
6.8. Validation of recommendations 85

7. Conclusion 93
Glossary 98
References 99
Appendix 1: Experiments conducted 107

Chapter 1

Introduction

1.1. Introduction

By accommodating problem solving and decision-making processes, simulation can be considered as a very powerful tool as it meets the demands of large-scale manufacturing systems identified by Law and McComas (1999). To be more competitive, manufacturing companies are forced into lower production costs, continuous quality improvements, lower capital costs, and the minimisation of risks. Businesses need to re-evaluate their existing manufacturing systems with the goal of re-organising or replacing their current systems. Real-world manufacturing systems are quite complex and are heterogeneous in nature, hence mathematical methods do not allow these systems to be evaluated accurately. As a consequence, Law and McComas (1999) identified simulation has become an invaluable tool for analysing their behaviour as it has the ability to deal with very complicated systems. Well known benefits of simulation include understanding complex interactions, identifying bottlenecks and the ability to perform "what if" analysis. Among the simulation techniques available, discrete event simulation software is proving to be highly favourable throughout the manufacturing industry discussed by Randell *et al.* (1999).

1.2. Simulation modelling of large and complex manufacturing systems

Manufacturing companies often develop models for a specific part of a larger manufacturing system, for example, a model for a cell. It appears, however, in certain situations that companies wish to model the entire manufacturing facility in order to evaluate the performance of this facility. Instead of building a new model for an entire system, it may be possible to amalgamate previously built models to create the required model of the whole system. However, difficulties arise when merging two independent stand-alone simulation models, due to one or more conflicts with variable names. It is a problem identified by the author, through discussing with simulation practitioners and software vendors. This problem has also been recognised by Hlupic (1999), and needs to be fully addressed. Modularity is the term used to define the construction of smaller,

independent and simpler simulation models, which are often referred to as 'modules'. These modules may be integrated to produce a single simulation model. As many simulation models become more complex, the need for modularity is increasingly being recognised by Decker (1999). Modularity has become a recognised technique to deal with these issues. These techniques will be discussed in chapter 2 in more detail.

1.3. Modular development of large scale manufacturing simulation models

In manufacturing companies, it is recognised in a survey conducted by Hlupic (1999) that the integration of two or more modules, which were constructed at different times, can be a difficult process. Simulation practitioners often only require the modification of a section or a single 'module' in order to deal with a specific problem. The time to alter a module could be significantly less than creating a new simulation model. Manufacturing companies will benefit greatly by adopting the modular approach. This can be shown in the diagram below (figure 1).

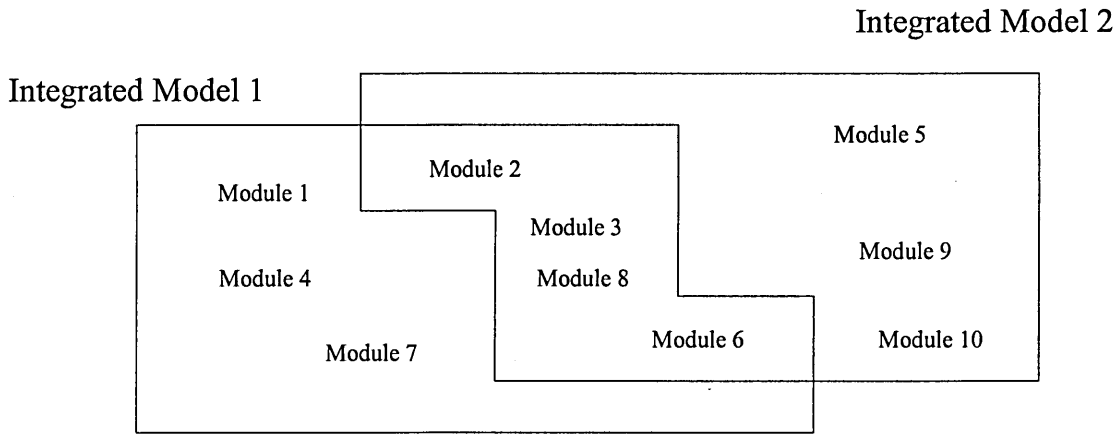


Figure 1 : Diagram showing the adaptation of modular approach.

1.4. Potential issues with modularity

There are potentially many issues surrounding modularity, some of which are listed below;

- Simulation packages may do particular tasks automatically i.e. referencing numbers generated by the simulation software. These reference numbers may not automatically change when the integration of modules takes place. Consequently, the reference number would be invalid.

- Data could be lost by deleting required building blocks during the integration process.

- The flow of entities through a manufacturing system may be misinterpreted during the integration of two modules.

- Misinterpretation of variable names i.e. the user may use the same resource name. This may not cause an error but might misinterpret the incorrect output as the resource utilisation increases.

1.5. The need for research

Ideally simulation practitioners would like to be able to 'cut and paste' modules – changing variable names, as required. This may not be the case.

The immediate question is:

"Would it be sufficient to bring together modules simply by changing the variable names?"

This is the basis of this research. As explained in section 1.2. it is an invaluable question which the manufacturing industry should consider.

The research hypothesis reflects on the construction of simulation models and analyses the difficulty, as well as identifying the issues of modularity of simulation models. The main objective of this research is to develop a framework to aid the user to minimise the error caused by the integration of simulation modules. In effect, a 'large' simulation model is to be built from 'smaller' components.

1.6. Contribution to the practice of simulation

This research will contribute to the practice of simulation with particular application to manufacturing industries. The benefits of adopting a modular approach are evaluated. The following points emerge;

- Changing the variable name alone is not adequate.
- A development of a set of recommendations will minimise the potential problems of integrating modules together.

1.7. Objectives of the project

The aim of this project is to develop a set of recommendations, which enables modular development of simulation models for large-scale manufacturing systems. This aim will be accomplished in the following four objectives:

1.7.1. Identify the issues involved in the simulation of large-scale manufacturing systems.

In order to understand the needs of simulation models, a survey of the current literature in the field of manufacturing systems and simulation will be conducted. The information accumulated on large scale manufacturing systems will be analysed and the problems encountered in simulation modelling of large-scale systems will be determined.

1.7.2. Construct a series of modular simulation models.

This task will consist of integrating a series of modules and constructing simulation models, based on real industrial situations. These models will be assembled individually as modules, and hence operated independently. Decker (1999), described how these individual processes are linked through 'interprocess communication'. The modules are connected in series, parallel or a combination of both. The simulation models will be constructed using Arena® simulation software, as described by Collins and Watson (1993). Figure 2 illustrates an example of a simulation model with modules and the interaction between them.

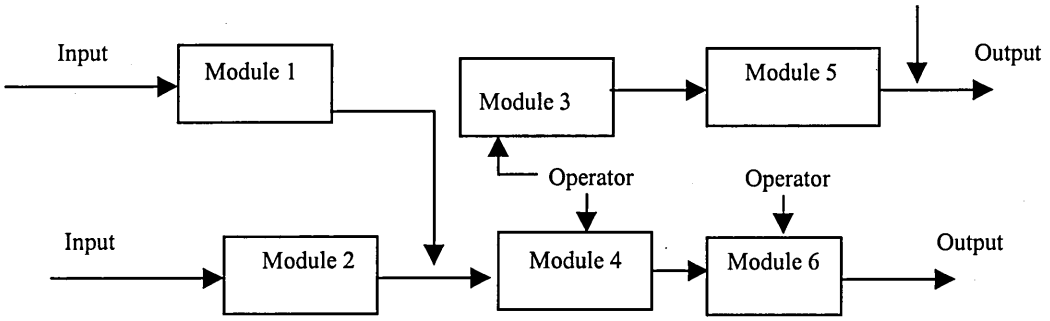


Figure 2 : Diagram showing the interactions between modules.

1.7.3. Study the issues involved with the integration of the modules developed in section 1.7.2.

Experience gained from the above stage will be used to assess the issues and determine the difficulty of integrating the simulation modules together. Maxwell (1999) described a methodology as a "specification approach method", where the modules will be encapsulated, and are only interfaced by their inputs and outputs. He believes that this method will bring great potential for modular model development in the field of simulation. Some of the current issues related to the integration of modules are:-

- Conflicts in variable names
- Interaction between modules (Inputs/Outputs)
- Redefinition of input variables (from individual modules to the integration of modules)

1.7.4. Develop a set of recommendations to enable the modular development of manufacturing simulation models.

The issues involved with the above task will be concluded. A set of recommendations to enable the modular development of manufacturing simulation models will be developed, this will ultimately minimise these imperative issues, if any, enabling any further developments, if necessary.

1.8. Delimitation of scope and key assumptions

To focus the scope of the project, there must be limitations. There are other areas in the field of simulation that can be researched i.e. healthcare, military and transportation. This project focuses, in particular, on large scale manufacturing systems. These boundaries include the following;

- Applied to a commercial simulation package.
- Discrete event simulation (discrete, dynamic and stochastic simulation models)

The important assumption made during this research is that the simulation modules are validated independently before the integration process takes place i.e. each module is independent and operates as an individual module.

Chapter 2

Literature Review

To identify the issues involved for the research programme, an in depth literature survey is necessary and the review considers the following topics in the field of simulation and related areas.

- Definition of simulation
- Discrete–event simulation
- Industrial benefits
- Drawbacks
- Product life cycle
- Modular simulation
- Work done in modular simulation
- Object-oriented simulation
- Distributed simulation

2.1. Definition of simulation

Simulation is a useful tool in mimicking reality; it validates decisions before physical implementation. Ball (1998) defined simulation as " the technique of building a model of a real or proposed system so that the behaviour of the system under specific conditions may be studied." As mentioned in section 1.1, manufacturing industries are forced to lower production costs and continuous improvements; this has led to flexibility in the use of automation and components. As a result, complex production systems need to be controlled in planning and manufacturing operations for optimum efficiency. However, simulation alone does not provide a solution; instead it provides a valuable tool for identification of existing or potential issues. Problems can be identified and the opportunity to test alternative solutions can be made possibly without interfering with the real system.

2.2 Different types of simulation models

2.2.1. Static and dynamic

Law and Kelton (1991) defined a static model as "...a representation of a system at a particular time, or one that may be used to represent a system in which time simply plays no role". These authors also give an example of a static model by Monte Carlo models.

A dynamic model defined by Banks (1999) "....simulation models represent systems as they change over time." He also gives an example of a bank starting at 9am and finishing at 4:30pm. The measures of performance can be described by a variety of issues; some of these are 'work in progress', 'throughput' and 'possible rejects'. A dynamic model gives more of a true representation of the real system by including queues (bottlenecks within the system), and delays. Animation incorporated in the simulation software can also be used to give a visual dimension to the working model.

2.2.2. Deterministic and stochastic

A stochastic simulation model defined by Banks (1999) "..has one or more random variables as inputs". This type of simulation model produces random realistic data as well as providing a true characteristic of a real system to the model. The author also relates this to the previous example of a bank; it would be classed as a stochastic simulation model by the random inter-arrival and random service times. A deterministic model is a simulation model that does not contain random numbers, where stochastic simulation models contain the probabilistic behaviour. Thus stochastic simulation models would be more suited to manufacturing systems.

2.2.3. Continuous and discrete

Harrell and Tumay (1997) defined discrete-event simulation as "change in state at discrete points define discrete-event simulation in time as a result of specific events". They also define continuous simulation as "model systems whose state changes continuously with respect to time". This research focuses on discrete-event modelling.

2.3. Discrete-event simulation

As aforementioned, refer to 1.8 this research concentrates on discrete-event simulation, focusing on the dynamic, stochastic and discrete simulation models. Banks (1999) defines discrete-event simulation as "modelling of systems in which the state variable changes only at a discrete set of points in time".

Simulation models in manufacturing systems are generally discrete-event. One particular interest of discrete-event simulation is the analysis of queuing systems. This can be especially useful if there is a bottleneck in a manufacturing system; an analysis of the work-in-progress of a simulation model, or the flow of entities within a manufacturing system.

Discrete-event systems can be either stochastic or deterministic. Stochastic simulation is concerned with random probabilistic components; dynamic simulation is related to a system represented within a given period of time whereby results are generated. Stochastic simulation gives a more realistic representation of a manufacturing system in comparison to a deterministic representation due to the randomness used; however, this can be disadvantageous by not giving a true representation of the system. This randomness gives an estimate of the manufacturing system being investigated.

Deterministic simulation gives a true representation of a system, as long as the parameters are correct. This randomness is called static representation; it is a situation where there is a set change in the dynamic behaviour in the use of a simulation language to compute the change of states at a particular event. The use of simulation language portrays the system more realistically than humans, not normally contemplated in terms of the changes in events and states. This is where a specific, purpose built simulation language becomes useful, particularly in the case of discrete-event distribution.

Discrete-event simulation can be broken down into key stages according to Tye (1999). Figure 3 showing this of the modelling process is shown below.

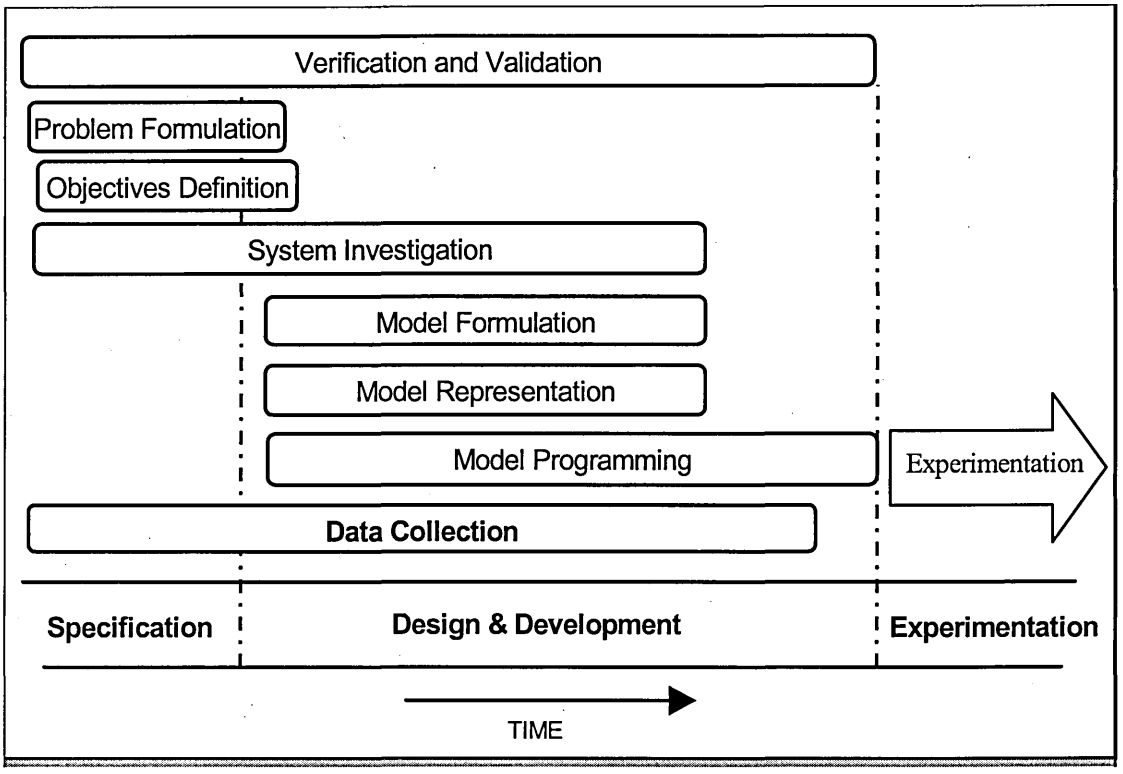


Figure 3 : Overview of the modelling process for discrete-event simulation

2.4 Trends in simulation

Simulation was first introduced in the 1962. Since then discrete-event simulation has become widely used by manufacturing industries. Complex systems have increased immensely over the past years; therefore, simulation tools have become more sophisticated, identified by Davis (1998). Simulation has enhanced the advancement in animation (e.g. Quest simulation software). This aids the user to visualise the systems' behaviour more easily. Commercial simulation software has also increased in power over the years. In its earlier stages, simulation models were only able to run on mainframe computers, and using only Fortran-based languages. As personal computers (PC's) have become more powerful over the last decade, simulation has become a proposition for remote computing, identified by Roberto (1997). According to Thomas Jefferson from the Intel Corporation (Banks, 1999), the characteristics of simulation packages, which must be improved, are 'speed, flexibility, ease of use, and accuracy.'

An example would be Ford and BMW companies (Kochan, 1998). Both manufacturing companies have identified that simulation plays an important part of the product life cycle in new product development.

Avni (1999) provided an example of reducing costs through 'seven wastes' in lean manufacturing. The author stated that after the 'seven wastes' have minimised the amount of waste and then simulation can be applied to quantify improvements. The use of simulation has expanded immensely and it is considered to be an accepted tool as a result of the following:

- Specific purpose of simulation languages
- Increase in computing capabilities.
- Increase in simulation methodologies.

The aim of simulation programs is to map objects from the real world on a 'one to one' basis. The simplest method of constructing these models is by using a programming language. Simula is an example. It possesses expressiveness, extensibility and reusability described by Wong *et al.* (1999). Due to increasing in demands for flexibility in manufacturing systems, objected-oriented modelling is gradually becoming more popular. It requires minimal effort for design modification. Garnett (1999) summaries and analyses a recent survey. The survey suggests that large portions of forty-one respondents are currently involved with simulation. A pie chart shown in figure 4 illustrates the responsiveness of the survey.

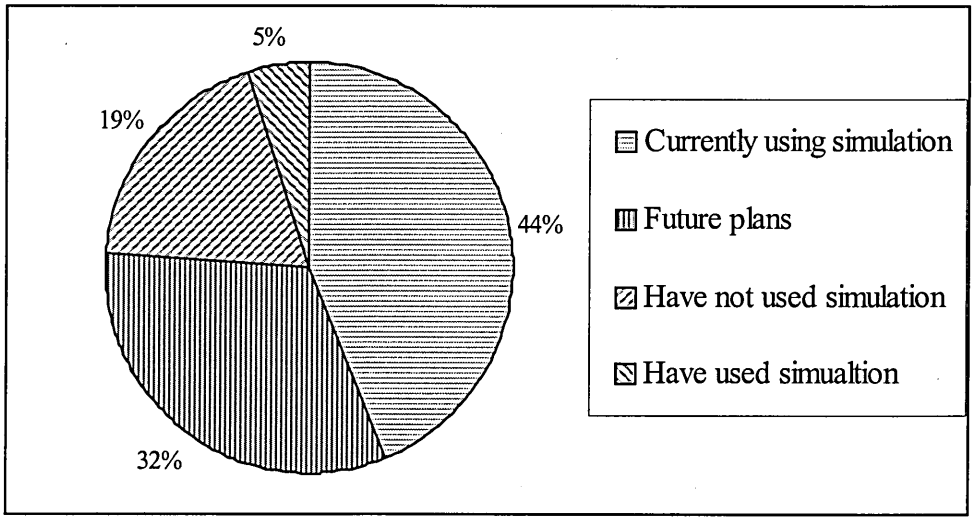


Figure 4 : Responsiveness of a survey conducted by Garnett (1999)

This pie chart suggests that simulation is very much "alive"; however, Garnett (1999) believes that simulation is a growing trend; he also believes simulation could be too complex to be mainstream.

2.5. Industrial benefits

In the recent years, simulation avoids disturbance by simulating the implementation of a machine before installations are implemented into a real system. The daily operations of a real manufacturing system are not disrupted by simulation. Robinson (1994) lists other benefits:

- Risk reduction
- Greater understanding
- Operating cost reduction
- Faster plant changes
- Answer 'what if' question
- Lead time reduction
- Capital cost reduction

2.5.1. Applications

The industrial benefits can be enormous if simulation is used in the correct manner. The initial cost may be high but this can be out-weighed by cost minimising methods resulting from a faster decision-making process. There are a variety of applications and benefits in the field of simulation. It can be applied to other areas, apart from manufacturing industries; nevertheless, they are the most popular applied application identified by Hlupic (1999).

Some other applications include:

- Healthcare (Strandridge, 1999)
- Military (Sisti, 1998)
- Transportation systems (Ruiz-Torres and Zapata, 2000)
- Education (Nance, 2000)
- Construction (Kamat and Martinez, 2000)

2.6. Issues in simulation

As computer hardware becomes more powerful, faster and easier to use over the years, manufacturing companies are recognising the potential usage of simulation by having the ability to simulate manufacturing situations using powerful computers. However, simulation also has issues that need to be addressed. Auguston (1997) suggested a comprehensive list of the "no-no's" in simulation, which are tabulated below (table 1),

1. Performing a simulation without clear definition of the objectives.
2. Believing that the model itself can compensate for data collection.
3. Lacking an understanding of statistical processes.
4. Failing to do order profiling.
5. Ignoring the effects of randomness.
6. Incorporating randomness inappropriately.
7. Failing to consider down time.
8. Making illogical assumptions.
9. Failing to question the results.
10. Failing to recognise that simulation is a study tool.

Table 1: Top ten list of simulation 'no-no's' (Auguston, 1997)

2.6.1. Issues: Limits in modelling capacity

Simulation is currently limited in modelling capability. Davis (1998) states that there are "critical deficiencies in the existing approaches to simulation modelling and analysis that limit both their efficacy and applicability in addressing these systems". He discussed three major concerns regarding simulation; modelling scope, model reusability, and model use; he also states that the user overestimates the performance of a model by simplifying the assumptions. This leads to errors in the model; therefore, validation at each simulation process is becoming essential. Banks and Gibson (1998) said that simulation needs constant validation of the simulation process. They state, "The expectations are often naive and sometimes unrealistic. If allowed to stand, unfulfilled expectations can spell failure for a simulation project"; often, the initial

design is validated. Banks and Gibson question the validity of the final design model simply because it is ignored, and the concept of validation is vital to the final design if misunderstood.

2.6.2. Issues: Validation and verification

One of the most difficult tasks facing a modeller is building an accurate representation of a real system, and to be used for decision-making. Decision-makers are concerned with the data derived from results of models, and subsequently use this information to make decisions. They (the decision-makers) often ask the question: "Are these results correct?" To assure these crucial decisions, two factors, verification and validation, have attempted to address the issues involved. Schlesinger *et al.* (1979) defined verification and validation by,

> Verification: "ensuring that the computer program of the computerized model and its implementation are correct,"
>
> Validation: "substantiation that a computerized model within its domain of applicability possesses a satisfactory range of accuracy consistent with the intended application of the model"

Carson (1986) suggested both processes should be involved from the start of the simulation project and be consistent throughout, except at the end. Therefore, it is vital that verification is employed throughout the simulation process.

In the validation process, there are vast amounts of techniques in testing the validity of simulation models. Sargent (1999), described 16 techniques that would be useful in determining the validation of simulation models. Some of these are listed below,

- Animation
- Degenerate tests
- Face validity
- Event validity
- Historical data validation

Sargent also mentioned numerous techniques for operational validity. This involves the output behaviour of the model, which must have the accuracy required consistent with operational specifications. He suggested three techniques to approach operational validity,

- Graphs of the model system data behaviour
- Confidence intervals
- Hypothesis

In addition Sargent (1999) stated that model verification is crucial and if the programmer uses high-level language then the technique carried out would be by the software engineer. Examples of techniques that might be used are object-oriented design, structured programming, and program modularity. Therefore, the only concern would be simulation functions, programming and correct implementation. There are two techniques for testing simulation software; namely static and dynamic testing. Examples of static testing techniques are 'walk-throughs' and 'correctness proofs'; examples of dynamic testing are 'traces' and 'internal consistency checks'. Validation of a simulation model is the crucial stage of the simulation process and costs the project extensively in time and money.

2.6.3. Issues: Incorrect data

The simulation process can be undermined by the use of bad assumptions and incorrect data. Auguston (1997) stated "without good input data, the chances of simulation success are slim". However, Carson (1986) concluded that the 'good' data is not always available. In large manufacturing companies, there are lots of data; but possibly in the wrong form: i.e. out of date or incorrect. When a constructed model is in use constantly or for a new purpose, then it should be noted that a system is continually changing. Therefore, data can be invalid for the system. Carson (1986) suggest re-validating the system at frequent intervals. This will ensure that the manufacturing system will be up to date with changes.

2.6.4. Issues: Inappropriate use of animation

Animation is also an aid to verify that the simulation model acts correctly with the real system. Law and McComas (1989) stated that the use of animation has many advantages. However, animation can also provide a 'false sense of security' about the

model presented to the client. Animation requires a lot of time to construct, which may defeat the object of the real problem of constructing the simulation model. Animation has also its advantages; the visual effect of animation can convince the management of a proposed alternative to a solution.

2.6.5. Issues: Modularity

Manufacturing companies have a tendency to build simulation models that represent sections of a manufacturing system and at a later stage it is often required to integrate these models together in order to evaluate a larger section of a manufacturing system. Issues arise due to the level of difficulty when integrating individual simulation models together. The issues raised could be potentially very difficult to resolve and time consuming. An example could be changing all internal reference numbers to a particular building block. Hence, the level of difficulty could be minimised by reducing the amount of issues arising and integration between simulation models could be less complicated.

2.7. Modularity and the modelling process

In the model building process, modularity plays a part in the following stages of development.

- Model formulation
- System investigation
- Model representation
- Model programming
- Validation and verification (V&V)
- Experiments
- Documentation

With reference to Tye (1999), there are nine stages to the model building process; modularity effects 78% thereof.

2.7.1. Model formulation

Tye (1999) stated that model formulation is decided on the following:

- How the system elements should be modelled?
- What level of detail they should be represented?

Modularity could play an important part in the level of detail of the simulation model.

2.7.2. System investigation

A system investigation decides what is included in the model. The modules of the system, and what is to be the function of each module determine modularity at this stage.

2.7.3. Model representation

Model representation is an important part of the modelling process cycle; it describes the infrastructure of the system in terms of modularity and prepares the information to develop the model on a computer. Modularity allows the planning of a large-scale model to be easily understood and planned, as it can be very complex to construct as one whole project. This representation allows preparation of the variables and limitation of the complex interactions of a large-scale simulation model.

2.7.4. Model programming

It is easier to program each small section than one whole simulation model. Modularity is very beneficial in determining the inputs and outputs of each module, as well as being easier to construct. This drives towards object-oriented simulation.

2.7.5. Verification and Validation (V&V)

V&V play a crucial part in the modelling life cycle, as noted in chapter 2. Modularity affects V&V by making the process of validation easier. Each module is verified and validated before integration. It is simpler to validate and verify each module rather than one large-scale model. One large-scale project has many variables to consider in comparison with each module. V&V often refer to a real time system as one that is constantly operating. Modularity can only affect certain parts of the system; therefore, it does not disrupt the system as it was validating it as a whole.

2.7.6. Experiments

This stage offers the possibility of expanding the simulation model. Modularity provides the capability of modifying a module, or a number of modules that do not affect the rest of the simulation model. As a result, potential problems may be easily solved through investigating and modifying individual modules.

2.7.7. Documentation

Documentation of the variables used also plays an important part for the user and other people, who wish to maintain and upgrade these models. Documentation will include the parameters, as well as the variables, used in each module.

2.8. Complex systems

In the past decade, manufacturing companies have been anxious to lower production costs as well as making improvements in quality. This is the underlying philosophy of 'Just In Time' (Storey, 1994). This leads to complex, large-scale manufacturing systems due to the increase in flexibility and automation. Some of these problems are much more difficult to address in large-scale manufacturing systems, where they are highly varied and complex. It is impractical to address the needs of all manufacturing systems with one solution or a software package. Clark (1996) stated that many manufacturing systems are impossible to analyse through simply thinking and proposing possible solutions due to the complexity of manufacturing systems.

In complex manufacturing environments, it is not practical to build the entire manufacturing system as a whole. A more feasible approach would be to develop individual 'units' and then integrate them together as a whole. As complex manufacturing systems increase, the importance of decision-making process intensifies; simulation can be beneficial in this process.

As mentioned in section 1.2, manufacturing systems are very complex and Nicol *et al.* (1999) stated that there is a gap between the size of simulation models and the scale of problems. Four issues are raised:

- How do we express a large-scale model?
- How do we validate a large-scale model?
- How do we solve a large-scale model?
- How do we trace and understand the output of a large-scale model?

The authors mentioned above concluded in their strategic directions in simulation research that it is more important to understand the results of a large-scale simulation, which has millions of interacting entities, with "interesting behaviour at levels of spatial and temporal scale". Joines and Roberts (1999) agreed that the 'real' limitation as they can foresee is the ability to represent complex systems and advocate discussion of the modelling style.

To accommodate this rapidly growing field of simulation, research is crucial in providing a more precise and accurate decision-making progress. The process of building simulation models can be time consuming and very costly; new techniques and research is required in order to reduce these effects.

2.9. Modular simulation

It appears that little research has been carried out concerning the issues surrounding the modular development of manufacturing simulation models with the use of a commercial package; although, certain methodologies have been carried out in hierarchical modular modelling within the field of discrete simulation (Proth *et al.*, 1995, Zeigler, 2000).

Pidd and Castro (1998) defined modular simulation, in terms of large-scale systems that are fragmented into smaller sub-systems, as 'developing modular approaches to reduce complexity of model building and to make such models easier to maintain. The authors suggested that the modular approach is the 'key' to cope with the complexity of large systems. Decker (1999) also suggested that modular simulation is the only way to satisfy all the clients' requirements, and sometimes, conflicting demands.

An approach to cope with complex dynamic systems is to use modular, hierarchical system modelling as stated by Praehofer (1996), which notably Ziegler (1984, 1987,

1990, 1993) introduced discrete-event simulation in the 1970's. According to Sargent (1993), there are three basic approaches to hierarchical modelling,

- Closure under coupling
- Metamodels
- "Specific software frame"

However, Sargent (1993) stated these methodologies as 'hierarchical modelling are not readily available' because encapsulation requires a different approach of modelling other than the "current practice" which simulation languages are not designed to execute.

2.10. Potential benefits of modularization for manufacturing industries

The proposed set of recommendations will enable simulation practitioners to plan the development of individual models and facilitate the subsequent integration. The approach will enable manufacturing companies to maintain individual modules more easily, and be able to alter a module easily, without affecting the rest of the integrated modules. The advantages for manufacturing companies include;

- Save time and cost
- Less prone to error
- Reusability
- Independence
- Multiple groups
- Maintainability
- Flexibility

Save time and cost

Once the simulation model is constructed, it will posses the advantage of being in a modular infrastructure. Modifying a particular module will be less time consuming and therefore less costly than constructing a completely new simulation model.

Less prone to error

It is easier to model a 'real-system' by breaking it up into sections rather than capturing it as a whole at once. Building and capturing smaller sections of the 'real-system' aids verification and validation. Consequently, the whole process is less error prone.

Reusability

Constructing large simulation models is often time consuming process. After the model has been built, it then has to be validated by constructing experiments to test the various different scenarios. Reusability of modules reduces the development time of applications. Therefore, the ability to re-use the model can be very beneficial.

Independence

As the complexity of applications increase, the need for a modular structure also increases. Highly independent modules are becoming necessary. This enhances the concept of encapsulation by minimising interdependencies between modules. Each module would be able to be verified and validated as it is integrated into a larger simulation model.

Multiple groups

Modularity facilitates team working. Each independent module can have a team working on it in parallel with other teams. Consequently, a simulation model can be built quickly in comparison to one person building a large-scale model. Communication between groups may provide valuable new ideas.

Maintainability

It will be easier to maintain and verify a particular independent module than it would be to alter a whole complex of simulation models. As large scale manufacturing systems change to meet new requirements, it will be relatively easy to re-configure and maintain the model by altering specific modules.

Flexibility

Modularity facilitates flexibility. Each module is highly independent and could therefore be changed to satisfy a different purpose. A modular infrastructure could readily adapt to changes in the manufacturing system.

Manufacturing systems have become so complex and diverse that a breakdown into smaller modules has become a requirement. Modularity will be most beneficial and applicable to large-scale simulation projects.

2.11. Work done in modular simulation

Modularity has been used in many areas of application and simulation languages; but a limited number of authors have applied modularity to a commercial simulation package, which has a simulation language with a simulator. Zeigler (1984, 1987, 1990, 1993) has discussed the methodology of the use of sub-models placed in a hierarchical structure in discrete-event systems. This consists of encapsulation, where sub-models are 'coupled' with other sub-models through the emphasis of inputs and outputs. However, modular simulation research has been carried out 'indirectly' to the related research and is concentrated in two areas. These two main areas are object-oriented simulation and distributed simulation. Figure 5 shown below shows how modular simulation using a commercial package fits into the research.

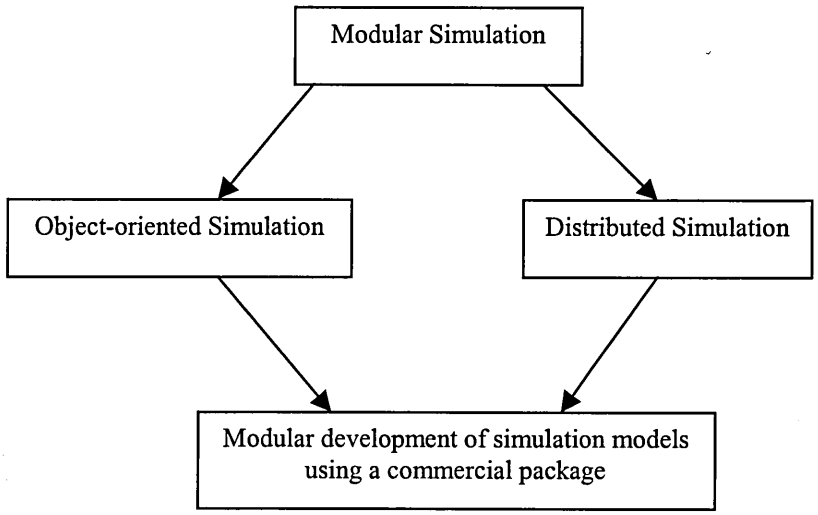


Figure 5: Where modular simulation using a commercial package fits into research

2.11.1. Object-oriented simulation

Modularity has been applied in object-oriented simulation (Roborts and Dessouky, 1998, Daum and Sargent, 1999, Hwang and Choi, 1999, Ninios *et al.*, 1995). It is chosen because each object is placed into sub-systems, and hence is in modular form as well as being encapsulated in the rest of the simulation program. Each sub-system communicates through their inputs and outputs to the rest of the program.

Object-oriented simulation technique is popular due to the benefits of maintainability, extensibility and reusability (Wong *et al.,* 1999). It has been noted by Narayanan *et al.* (1998) that one fundamental problem exists when applying simulation languages to manufacturing systems; the authors state this as abstractions used to describe the system being analysed. This rapid growth is appealing because it provides a more 'natural mapping' paradigm than the traditionally used concept of the 'seize-hold-release' paradigm, as stated Narayanan *et al.* (1998). This major change in concepts has the ability, in manufacturing systems, to map 'one to one' between objects. The key principles of object-oriented simulation are based on the three concepts of classes, inheritance, polymorphism and encapsulation.

Narayanon *et al.* (1998) discusses five different research areas to implement object-oriented simulation:

- BLOCS/M
- DEVS
- Lavel
- OOSIM
- OSU -CIM

As noted by Narayanon *et al.* (1998), each one of architectures mentioned above, a strategy is also discussed to implement object oriented simulation; however, according to these authors, they have isolated two distinct phases: development and testing. The development of the architecture (figure 6) begins with a domain. At this point, a process and modelling takes place which, results in the proposed architecture ('...a generic manufacturing system modelling formalism...'). Using the traditional simulation process, testing of the proposed architecture and all architectures has been proven to be

workable. Narayanan *et al.* (1998) also notes that the testing phase shows opportunities for better optimisation of architecture.

Table 2 shows the research of all the architectures of object-oriented simulation (page 23) and Narayanan *et al.* (1998) have identified the research objectives with justification of these objectives. Figure 6 shows the development of the architecture, which is used to create and analyse specific simulation models. The implementation and applications are linked to the development of the architecture, which demonstrates and evaluates the classes and methods. Table 2 shows the research objectives for object-oriented simulation and the justification for each system or group. Figure 7 and 8 shows the different structures and abstractions for the different manufacturing entities of each architecture. In these structures, two fundamental issues are discussed; the first issue is to discuss the representation of a specific behaviour and the way the architecture of separate structures are coupled. They conclude in the paper that object-oriented simulation can be very beneficial but the design principles and domain analysis must be further developed to ensure maximum benefits. These authors believe that the coupling mechanism is a weak point in object- oriented programming and need further development.

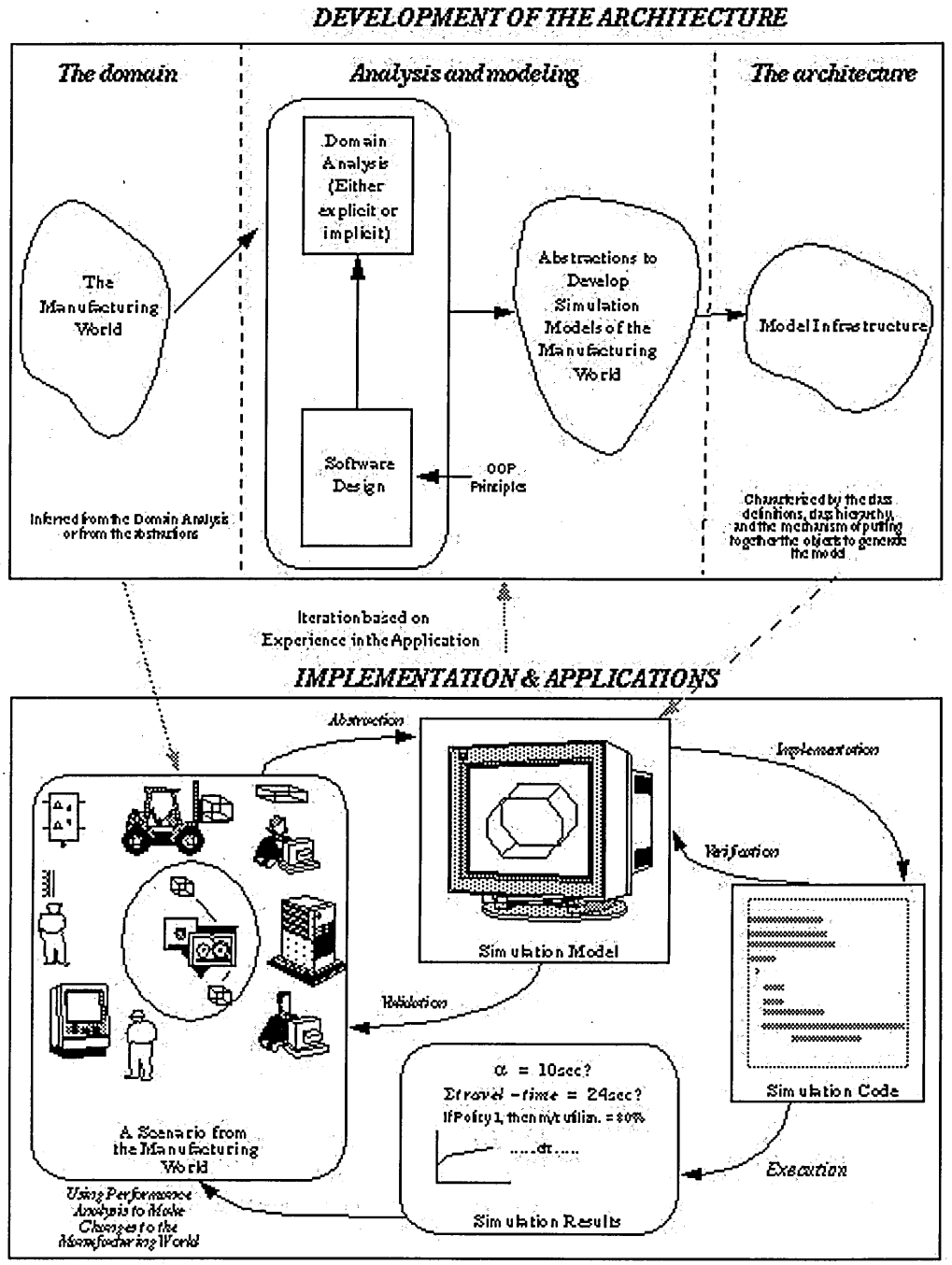


Figure 6 : 'Activities associated with object-oriented manufacturing simulation research' (Narayanan *et al.*, 1998)

System/Group	*Research Objectives*	*OOP Rationale*
BLOCS/M	Design a library of software modules to assemble special-purpose simulation models for manufacturing. Make the class library more reusable and easily comprehensible. Develop simulation models so they run efficiently.	Reusability. Ease of maintenance.
DEVS	Hierarchical and reusable model bases. Combining simulation modeling and AI techniques. Exploring distributed simulation models and architectures.	Exploring compatibility between OOP and discrete-event world-view. Reusability.
Laval	Develop an intelligent object-oriented model and simulation of manufacturing systems. Simplify the description of complex systems.	OOP provides a hierarchical world-view and polymorphism. Natural mapping.
OOSIM	Develop an object-oriented simulation modeling framework for representing the interactions between parts, automated processes, and operator problem solving for discrete manufacturing systems. Design a reusable library of classes to support modeling of manufacturing systems at different levels of abstraction from the viewpoint of material flow control and supervisory control. Support real-time, interactive simulations.	Natural mapping Reusability.
OSU-CIM	Develop a modeling, analysis and optimization environment for manufacturing systems. Develop a modeling framework that permits the separate specification of physical, information and control elements. Develop formal methodologies for multi-level modeling and simulation model parallelizing. Design and implement an OOP-based modeling environment that permits programming-free model creation and multiple problem-solving approaches with a single base model.	Modularity and reusability. OOP facilitates modeling at different levels of abstraction. Natural mapping.
SmartSim/ SmarterSim	Produce a simulation environment that can be used by manufacturing engineers as a computer-aided design tool for the design of manufacturing systems.	OOP is useful in creating a simulation program generator. A good mapping is possible between system entities and icons in the software. Reusability.

Table 2: 'Research objectives and rationale for using object-oriented programming' source (Narayanan *et al.*, 1998)

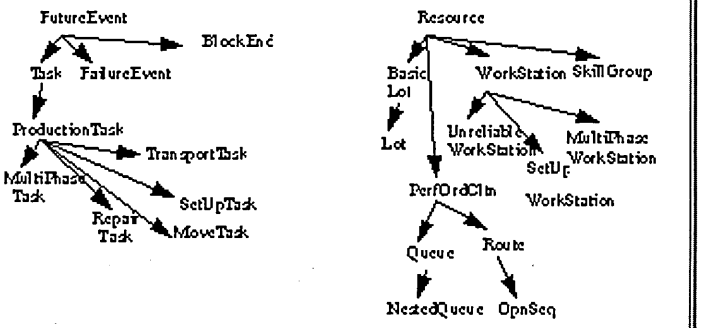
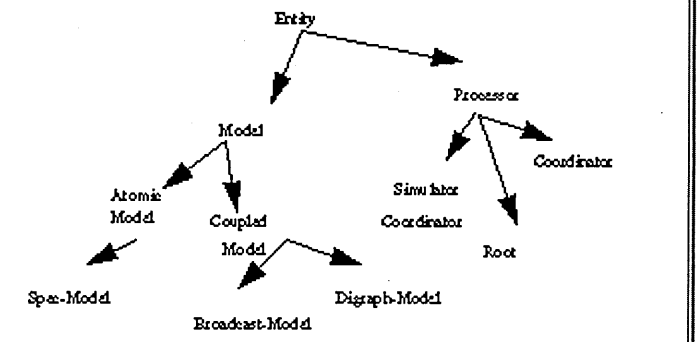
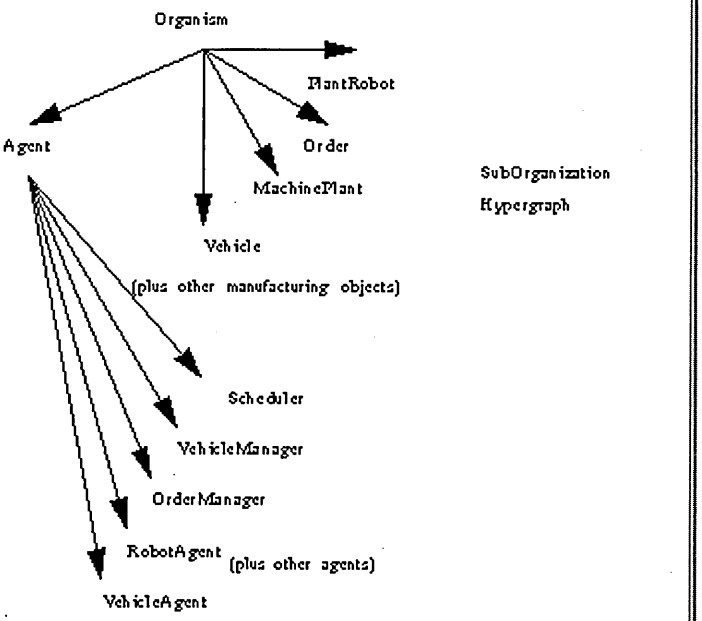
System/Group	Manufacturing Classes & Hierarchy	
BLOCS/M		
DEVS		
Laval		

Figure 7: 'Manufacturing classes and hierarchy' (Narayanan *et al.*, 1998)

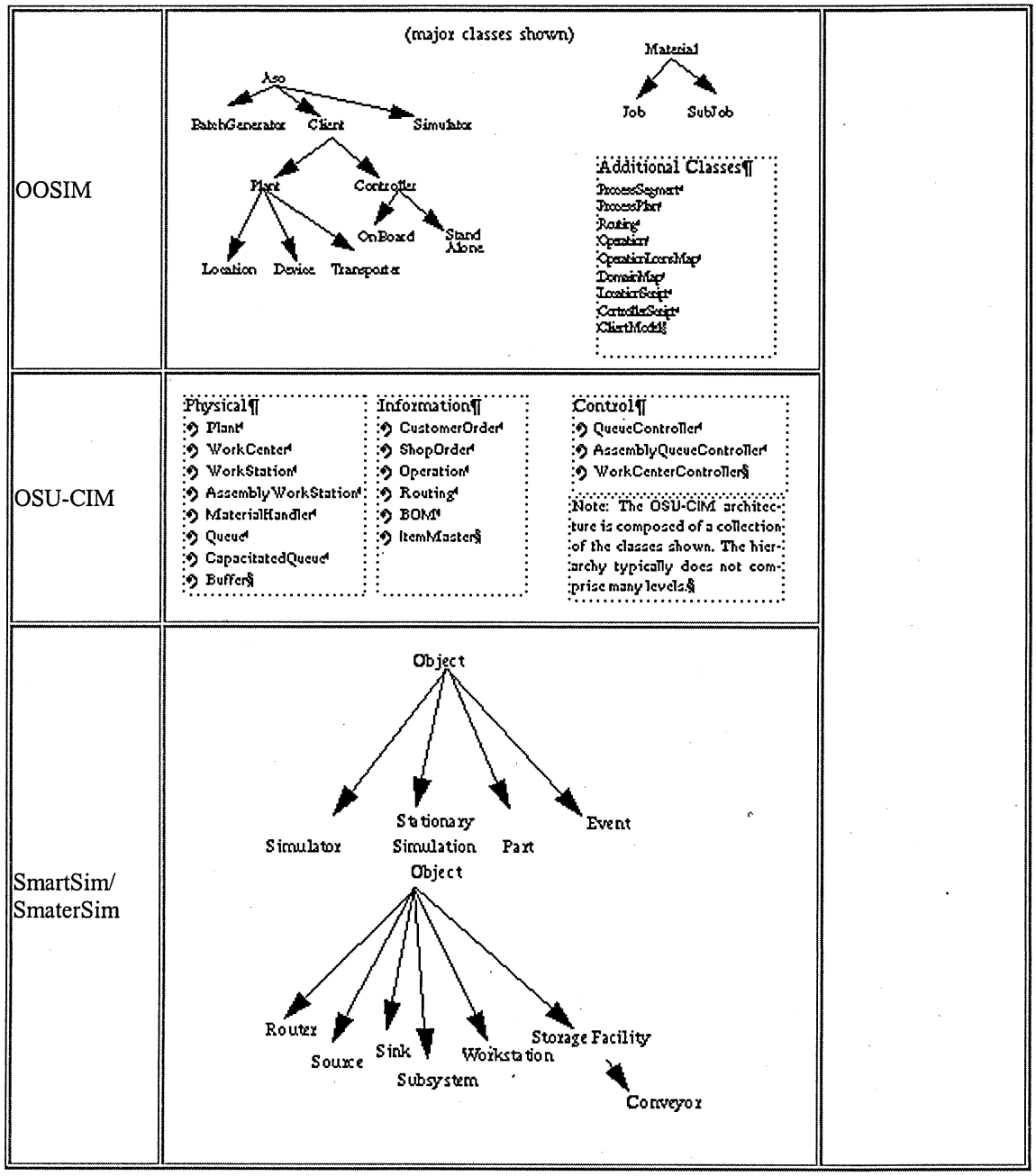


Figure 8: 'Manufacturing classes and hierarchy' (Narayanan *et al.*, 1998)

Luna (1991) described modular concepts applied to object-oriented environment. He discuses two advantages, firstly, each task is clearer as it is in a modular form and secondly, each part can be tested and developed incrementally. Sargent and Daum (1999) mentioned object-oriented simulation and hierarchical structures and conclude that for ease of manageability, the overall problem can be broken up into smaller problems. This leads to inter-operability.

However, Ball (1998) stated that the bridge between object-oriented techniques and traditional software design is still far off. Traditional software has been designed so that events and data have been used throughout the software but in the object-oriented approach, data and events are been grouped together. An example of software that uses object-oriented simulation is Simple ++.

However, Ülgen (2000) viewed that there should be no 'hard coding' required for object-oriented simulators because it is difficult to learn but recommends a 'menu driven' simulation software that captures all the true characteristics of object-oriented simulation.

Meinert *et al.* (1999) provided an example of a modular system: the United States Postal Service (UPS), where, the modular system evaluates the material handling system. Commercial simulation packages, based on a simulation language, have less flexibility in modularity. These authors have also mentioned the use of a hierarchical structure with the incorporation of sub-models.

2.11.2. Distributed simulation

Distributed simulation, also known as parallel simulation involves modularity. The basic concept of this technique is the execution of discrete-event simulation programs, where it is executed on a multiprocessor system or a network of workstations. Cavitt *et al.* (1996) stated that distributed simulation is a very cost-effective technique to examine, and the ability to understand complex real world systems.

Distributed simulation can bring many benefits. It reduces the simulation run time by taking advantage of multiple processors working in parallel. Other advantages of this technique mentioned by Fujimoto (1999) are,

- Geographical distribution
- Execution of integrating simulators from different manufactures
- Fault tolerance

Luna (1992) discussed a scheme called a "coupling scheme". This consists of each module being interfaced with the inputs and outputs. They argued that each module must be regarded as a "black box" and that the only communication to the mechanics in the black box are through the inputs and outputs. They also stated that each module must be independent.

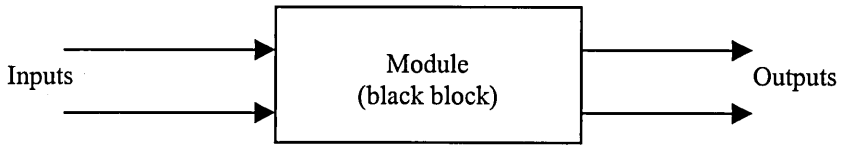


Figure 9: Inputs and outputs can only communicate to black box. (Luna, 1992)

Davis (1999) argued from the researchers point of view that the initial development of distributed simulation was to cope with large models but, as stated in section 2.4., computers have increased in power and memory. Hence, the need for research to address these issues has declined. However, previously, the Defense Modelling and Simulation Office (DMSO) of the Department of Defense (DoD), U.S.A. attempted to develop a framework that allowed individual models to operate together in a situation, called High Level Architecture (HLA), using distributed simulation applications. The advantage of this technique is that it facilitates the reuse and interoperability of simulations. Davis and Moeller (1999) listed some of the advantages for HLA,

- Maximise the reusability of existing simulation models.
- Allow individual simulation models to be integrated in order to model more complex systems.
- Allow the individual simulation models to interact that support distributed simulation technologies.

Davis (1996) described that HLA is a technique, which saves simulation models from previously constructed simulation models, which are not used again, once the problem has been solved. Hence, simulation models will be used to their maximum potential, but this does not 'promote the future' of simulation and states that it is not the ideal framework for future simulation needs. John Carson from Autosimulations states that the HLA is too complicated to develop unless it is simplified and 're-jigged'.

Moreover, McLean and Riddick (2001) stated with reference to software architectures that issues of integration problems encountered by software vendors and simulationists need to be addressed.

2.12. Conclusion

It is evident that many authors have addressed in many diverse areas, but it appears that only one journal (Meinert *et al.*, 1999) has addressed the issues surrounding commercial packages, which have a simulation language content as well as a simulator.

Therefore, research in the modularity of simulation models using a commercial package is required in order to reduce the gap between research in object-oriented simulation, distributed simulation and real manufacturing systems. A more detailed hypothesis (a refinement of section 1.5.) can be stated, namely

"Would it be possible to create an integrated simulation model from independent simulation modules simply by changing the variable names?"

Chapter 3
Methodology

3.1. Introduction

This chapter endeavours to describe the research work undertaken in order to address the issues identified in chapter one. The chapter begins with a brief introduction to various software packages that are currently available on the market followed by a justification of software package selected for this research. Also included in this chapter is a detailed analysis of the simulation software package, namely Arena.

Nevertheless, the ultimate objective of this chapter is to elaborate on the research work carried out. Thus, analysis of each phase of the research process is embarked upon. There are a total of five phases of research work undertaken, namely; analysis of Arena building commands (phase 1), elimination of logic blocks (phase 2), experimentation (phase 3), recommendations (phase 4) and conclusion (phase 5). Each phase of the research procedure is based on Arena. Finally, the chapter ends with a detailed example of an experiment conducted.

3.2. Selection of software

There are many packages available on the market that could be used for this analysis, some examples are Arena (Sadowski *et al.,* 1998, Arena Internet web page, 2000, Swets and Drake, 2001, Kelton W.D. *et al.*, 1998), Automod (Phillips, 1998, Stanley, 2000), Quest (Barnes, 1997, Mahajan *et al.*, 1993, Quest Internet web page, 2000), Witness (Rawles, 1998, Witness Internet web page, 2000), Extend (Krahl, 2001, Extend Internet web page), Simul8 (Simul8 Internet web page, 2000).

There are many different representations of simulation software but resource based and entity flows are the most popular. Resource based is represented by each resource defining their individual inputs and outputs. Entity flow is represented by a series of building blocks to determine the flow of each entity.

The diagram (figure 10) below illustrates the two major types of representations:

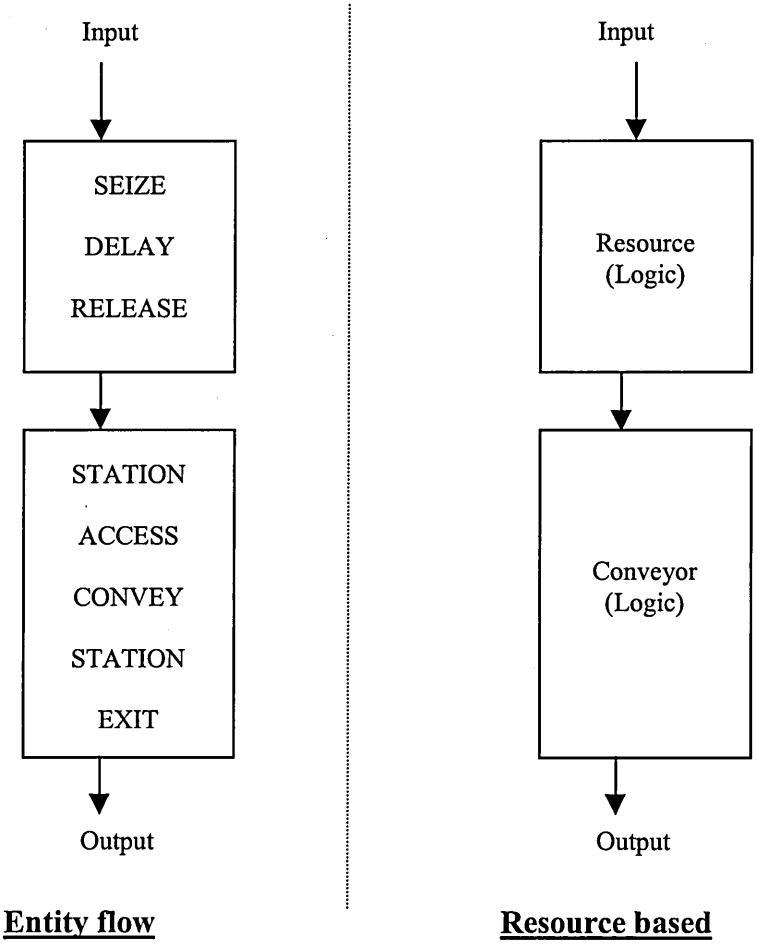


Figure 10: Types of representations

Some of the well-established simulation software is listed below illustrating the categorisation (table 3).

Simulation software	Resource based	Entity flow based
Witness	X	
Quest	X	
Simul8	X	
Arena		X
Automod		X
Promodel		X

Table 3: Categorisation of well-established simulation software packages

For the analysis, a simulation package called Arena (Collins and Watson, 1993, Sadowski *et al.,* 1998, Arena Internet web page, 2000, Swets and Drake, 2001, Kelton W.D. *et al.*, 1998) was selected. The justification for using Arena is listed below:

- Well-established commercial product and one of the most widely used software package.
- In-house expertise (within Sheffield Hallam University).
- Good technical support
- Flexible to suit many industries
- Easy to use
- Previous experience of the author

Arena is a highly recognised simulation package that has been used by large corporations. Companies who utilise the Arena package (Kelton *et al.,* 1998) have found the package to be an invaluable tool. Even though Arena package was used for this research, yet the research and outcomes can be generalised to suit for other simulation packages.

3.3. Arena simulation package

Arena® provides flexibility for a broad range of applications. The package is based on a simulation language known as SIMAN. The package requires input parameters (data) as well as the logic in order to create a simulation model. Consequently, the user requires only a minimal knowledge of programming languages. Some of the targeted markets include; manufacturing, call centres and business processes.

Arena also has the ability (Professional edition) to create customised building blocks. This provides flexibility to the user. Related subjects can be grouped together as libraries within templates.

Structure of Arena software

Arena is based on simulation language called SIMAN, which was developed in 1981 and offers a range of building blocks. These building blocks carry out a specific task. They are grouped together into five templates i.e. common, support, transfer, blocks and

elements. Arena has three levels of templates. The lowest level template used in Arena is the elements and block templates. These are the basic statements of SIMAN, and are very useful for complex algorithms or loops. Block and element templates are the lowest levels of logic and data modelling in Arena. Block modules define the logic and characterisation of different objects that are used in simulation. The element module becomes useful when additional information is required that is not represented by higher-level modules i.e. tallies and frequencies.

Higher levels of templates are the support and transfer. The building blocks within these templates are more customised to the user. This makes the building block easier to use in comparison to the block and element templates.

The highest-level template in Arena is the common template. High-level template in commercial simulation packages caters for common processes or a collection of processes. They are therefore less flexible to the users' requirements but very user friendly. This is illustrated in figure 11.

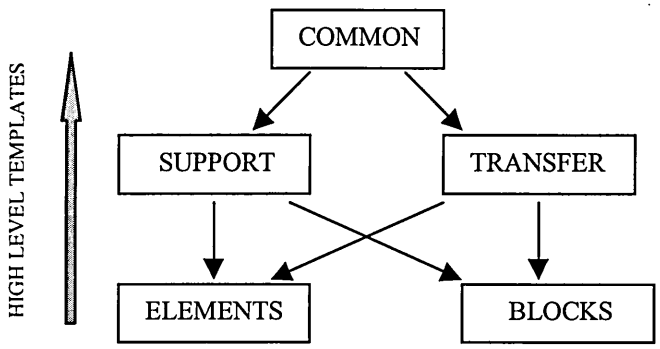


Figure 11: Template structure of Arena simulation software

The diagram (figure 12) below outlines the procedure of the experiments.

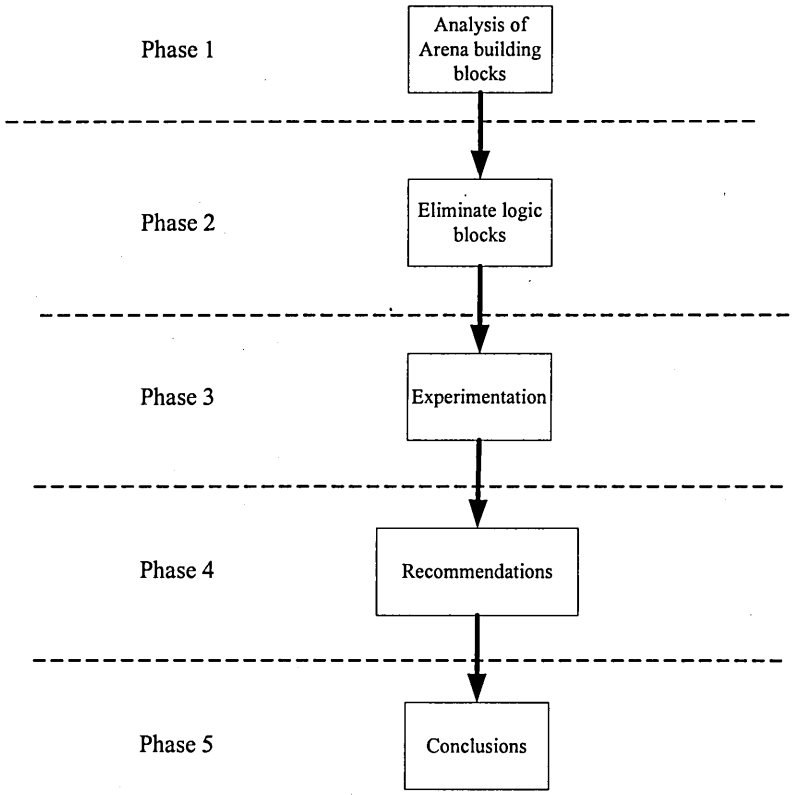


Figure 12: Outline of different phases of experiments conducted

Details of each individual phase will be described in chapter 4.

3.4 Conclusion

In the attempt to address issues raised in chapter one, the author anticipated to lay the foundations for this research project through in-depth investigation of potential resources and knowledge currently available. The research process provides a good insight to the simulation software Arena, which will prove to be invaluable to the experimentation at the next stage. According to the research undertaken, Arena appeared to be very user friendly and highly suitable for the purpose of the experimentation to be undertaken.

A discussion of the step by step process of the experiments conducted is presented in the following chapter.

Chapter 4
Experimentation

4.1. Introduction

This chapter endeavours to describe the research work undertaken in order to address the issues identified in chapter one. The ultimate objective of this chapter is to elaborate on the research work carried out with a detailed example of an experiment conducted. Thus, analysis of each phase of the research process is embarked upon.

There are a total of five phases of research work undertaken, namely; analysis of Arena building commands (phase 1), elimination of logic blocks (phase 2), experimentation (phase 3), recommendations (phase 4) and conclusion (phase 5). Each phase of the research procedure is based on Arena.

4.2. Phase 1: Analysis of Arena building commands

Arena building blocks generally consist of a logic element and a data element. However, they can be characterised into three groups;

- Logic only
- Data only
- Logic and Data

The logic element is defined by Arena software as;

"....Logic modules are connected together to define the process through which entities flow (customers, work pieces, patients, communication packets, etc.) During the simulation run, entities may arrive at and depart from logic modules that remain dormant until they are activated by the arrival of an entity...."

Source: Arena On-Line Help

The Arena software also defines the data element as:

".....Data modules are used to define data associated with the model. Unlike logic modules, data modules are not connected to other modules. Entities do not arrive at or depart from a data module. Data modules are passive in nature and used only to define data associated with the system....."

Source: Arena On-Line Help

Often building blocks are constructed with a combination of logic and data elements. This is to provide the user a more useful building block.

In addition, each building block belongs to one of eleven different categories, namely:

- Entities and attributes
- Station
- Resources
- Queues
- Storages
- Sets
- Transporters
- Conveyors
- Sequences
- Statistics
- Run control

In order to determine the appropriate section for each building block, the following steps are taken:

1. Insert a building block from one of the five templates into an active window, shown in figure 13. In this case, the building block inserted into the active window is ASSIGN from the support template.

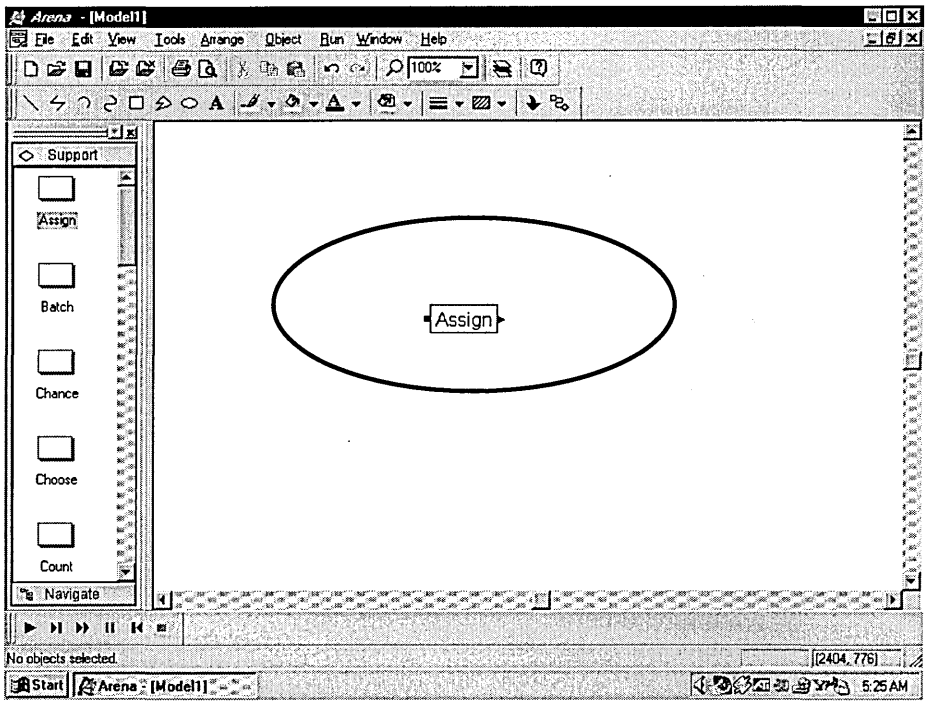


Figure 13: Insertion of building block from templates

2. Make the building block active and insert the required information in all the parameters inside the building block (shown in figure 14). The information inserted is a variable name called dummy, with a value of 1.

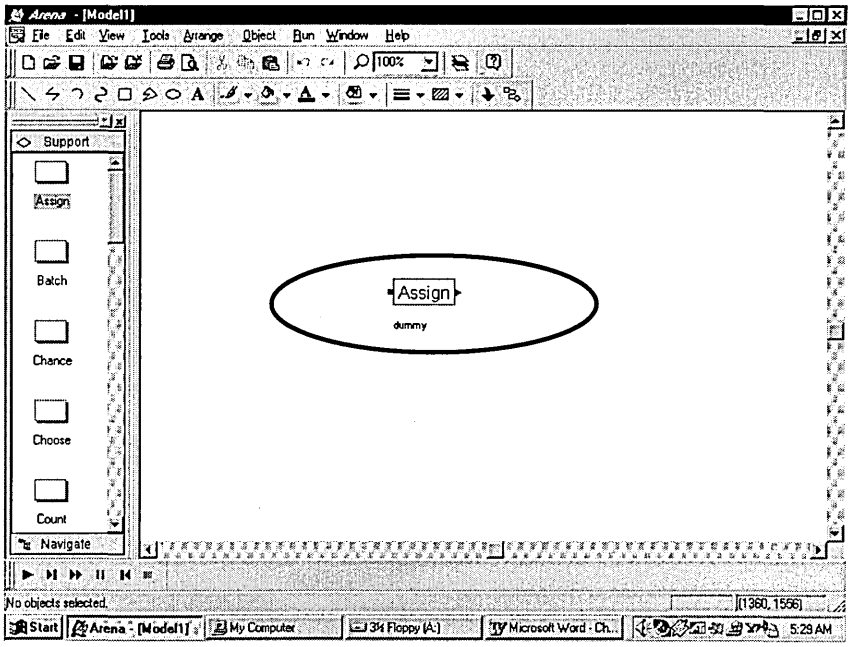


Figure 14: Input variable name

3. Go to the run menu, SIMAN and then view command (shown in figure 15). This is to view the basic SIMAN commands.

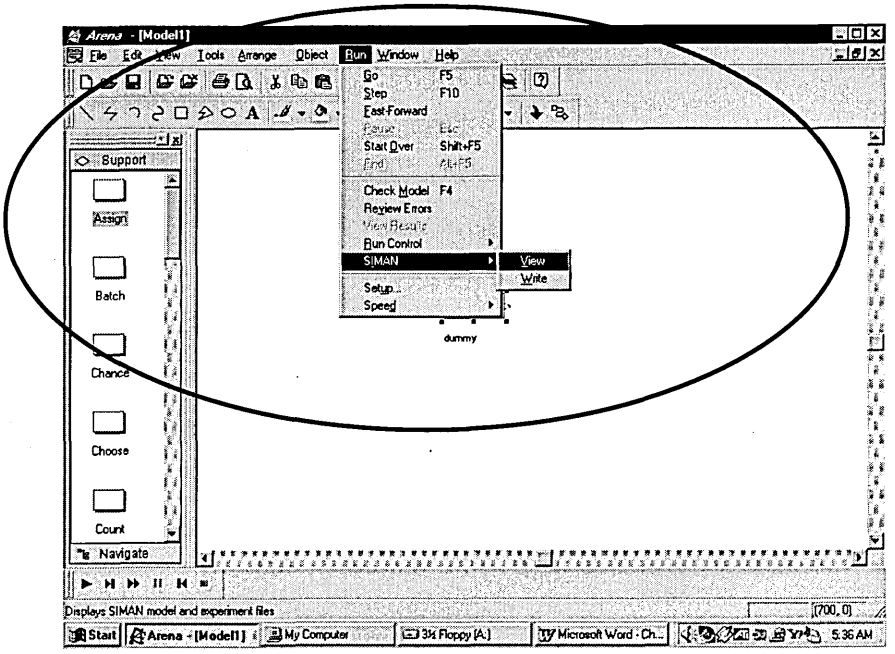
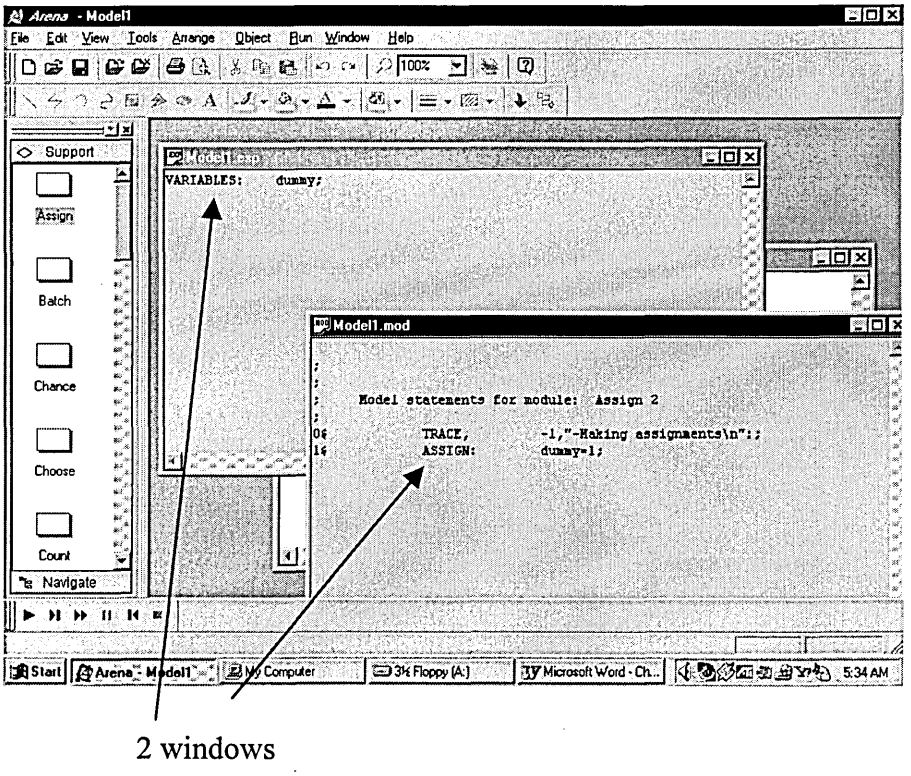


Figure 15: Viewing basic SIMAN commands

4. Two windows will appear regarding the building block (shown in figure 16).



2 windows

Figure 16: Pop up windows following insertion of building block

5. One window will show data information with a extension filename called *.exp and the other window is a logic window, with an extension filename *.mod (shown in figure 17)

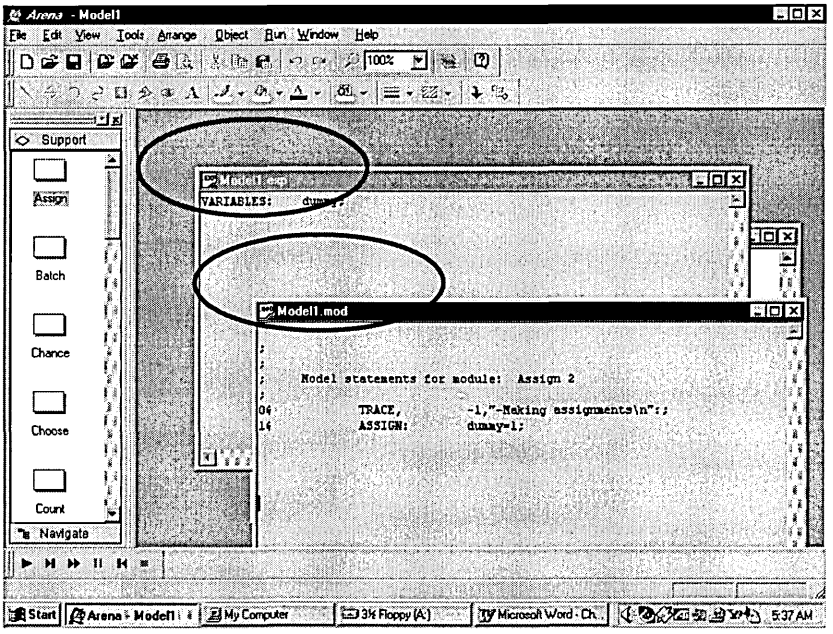


Figure 17: Extension files *.exp and *.mod

6. The SIMAN language scripts that are written inside these windows determine the building blocks function. The information written (SIMAN) in these windows determine the characteristics of the building block (shown in figure 18).

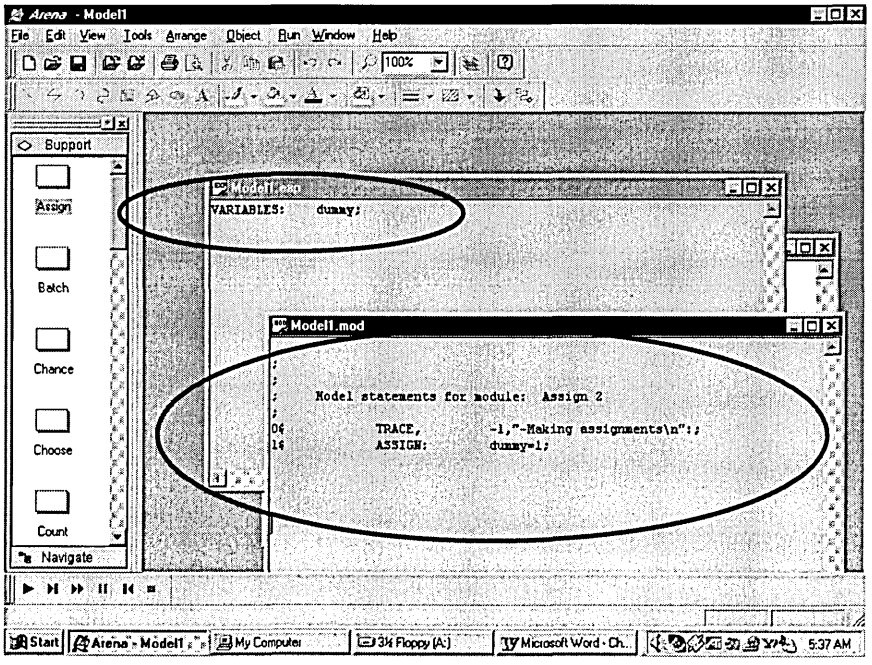


Figure 18: Determination of building block characteristics

An example of an Arena command that only contains logic properties is a CHANCE building block. To illustrate this further a few building blocks examples are shown below (table 4);

	Template	Logic only	Data only	Logic and Data
ROUTE	Transfer	X		
ASSIGN	Blocks	X		
RELEASE	Blocks	X		
RECEIPES	Common		X	
SETS	Support		X	
STATIONS	Element		X	
ADVSERVER	Common			X
SIGNAL	Support			X
FREE	Transfer			X

Table 4: Examples of building blocks

Certain Arena commands can be eliminated from the analysis, as they are not internal commands, but external commands. These building blocks are blacked out to show these building blocks are not included. An example of an external building block would be the WRITE building block in the support template.

The results of the experimentation with the remaining building blocks are shown below (table 5 – 9);

Common Template

	Entities and attributes	Station	Resources	Queues	Storages	Sets	Transporters	Conveyors	Sequences	Statistics	Run Control	Variables	Data / Logic or Both
Statics	X												DATA
Depart	X									X			BOTH
Receipes	X	X											DATA
Arrive	X	X				X							BOTH
Sequences	X	X							X				BOTH
Actions		X											BOTH
Leave		X					X	X					LOGIC
Process			X				X						BOTH
Enter			X			X							BOTH
Resources			X							X			BOTH
Server	X	X	X				X	X					BOTH
Inspect	X	X	X				X	X					BOTH
Advserver	X	X	X	X		X	X	X		X			BOTH
Containers										X			BOTH
Menu											X		BOTH
Statistics			X	X			X	X					DATA
Sets		X	X	X	X					X			DATA
Queue				X									DATA
Storage						X							DATA
Simulate											X		DATA
Expressions												X	DATA
Variables												X	DATA
Animate													

Note:- The building blocks that are blacked out are commands that communicate outside the Arena environment and are irreverent to the research.

Table 5: Results from experiments conducted (common template)

Support template

	Entities and attributes	Station	Resources	Queues	Storages	Sets	Transporters	Conveyors	Sequences	Statistics	Run control	Variables	Data / Logic or Both
Assign	X												BOTH
Batch	X												LOGIC
Chance	X												LOGIC
Choose	X												LOGIC
Create	X												BOTH
Dispose	X												LOGIC
Duplicate	X												LOGIC
Match	X												BOTH
Read	X												BOTH
Signal	X												BOTH
Split	X												LOGIC
Tally	X												BOTH
Wait	X												BOTH
PickQueue		X											LOGIC
PickStation		X											BOTH
Station		X											BOTH
Delay			X										BOTH
Release			X										BOTH
Seize			X										BOTH
Store					X								BOTH
Unstore					X								BOTH
Count												X	BOTH
Write													

Note:- The building blocks that are blacked out are commands that communicate outside the Arena environment and are irreverent to the research.

Table 6: Results from experiments conducted (support template)

Transfer Template

	Entities and attributes	Station	Resources	Queues	Storages	Sets	Transporters	Conveyors	Sequences	Statistics	Run Control	Variables	Data / Logic or Both
Route		X											LOGIC
Route Path		X											DATA
Activate							X						BOTH
Allocate							X						BOTH
Free							X						BOTH
Halt							X						BOTH
Move							X						BOTH
Network Link							X						DATA
Request							X						BOTH
Transport							X						BOTH
Transporter							X						BOTH
Access								X					BOTH
Access								X					BOTH
Convey								X					BOTH
Conveyor								X					BOTH
Distance								X					DATA
Exit								X					BOTH
Segment								X					DATA
Start								X					BOTH
Stop								X					BOTH

Table 7: Results from experiments conducted (transfer template)

Blocks template

	Entities and attributes	Station	Resources	Queues	Storages	Sets	Transporters	Conveyors	Sequences	Statistics	Run control	Variables	Data / Logic or Both
Combine	X												LOGIC
Copy	X												LOGIC
Create	X												LOGIC
Delay	X												LOGIC
Dispose	X												LOGIC
Dropoff	X												LOGIC
Duplicate	X												LOGIC
Proceed	X												LOGIC
Qpick	X												LOGIC
Split	X												LOGIC
Unblock	X												LOGIC
Scan		X											LOGIC
Station		X											LOGIC
Transport		X											LOGIC
Alter			X										LOGIC
Assign			X										LOGIC
Preempt			X										LOGIC
Release			X										LOGIC
Seize			X										LOGIC
Select			X										LOGIC
Signal			X										LOGIC
Insert				X									LOGIC
Match				X									LOGIC
PickQ				X									LOGIC
Pickup				X									LOGIC
Queue				X									LOGIC
Request				X									LOGIC
Search				X									LOGIC
Wait				X									LOGIC
Group					X								LOGIC
Store					X								LOGIC
Unstore					X								LOGIC
Free						X							LOGIC
Halt						X							LOGIC
Activate							X						LOGIC
Allocate							X						LOGIC
Capture							X						LOGIC
Move							X						LOGIC
Relinquish							X						LOGIC
Remove							X						LOGIC
Route							X						LOGIC
Access								X					LOGIC

Convey								X				LOGIC
Exit								X				LOGIC
Start								X				LOGIC
Stop								X				LOGIC
Block									X			LOGIC
Branch									X			LOGIC
Tally											X	LOGIC
Trace											X	LOGIC
Count												LOGIC
Detect												LOGIC
Findj												LOGIC
Read												LOGIC
Write												
Begin												
Close												
Else												
Elseif												
Endif												
Endwhile												
Event												
If												
Include												
Modifiers												
VBA												
While												
Zap												

Note:- The building blocks that are blacked out are commands that communicate outside the Arena environment and are irreverent to the research.

Table 8: Results from experiments conducted (blocks template)

Elements Template

	Entities and attributes	Station	Resources	Queues	Storages	Sets	Transporters	Conveyors	Sequences	Statistics	Run Control	Variables	Data / Logic or Both
Arrivals	X												Data
Attributes	X												Data
Discrete	X												Data
Nicknames	X												Data
Sequences		X											Data
Stations		X											Data
Failure			X										Data
Resources			X										Data
Schedules			X										Data
Statesets			X										Data
Queues				X									Data
Rankings				X									Data
Storages					X								Data
Sets						X							Data
Distances							X						Data
Intersections							X						Data
Links							X						Data
Networks							X						Data
Redirects							X						Data
Transporters							X						Data
Conveyors								X					Data
Segments								X					Data
Counters										X			Data
Cstats										X			Data
Dstats										X			Data
Reports										X			Data
Outputs											X		Data
Project											X		Data
Replicate											X		Data
Tasks											X		Data
Trace											X		Data
Begin												X	Data
Expressions												X	Data
Frequences												X	Data
Initialize												X	Data
Levels												X	Data
Parameters												X	Data

Rates												X	Data
Recipes												X	Data
Reportlines												X	Data
Seeds												X	Data
Statics												X	Data
Tables												X	Data
Tallies												X	Data
Variables												X	Data
Blockages													
Continuous Distributions													
Events													
Files													
Include													
Pictures													
Rules													

Note:- The building blocks that are blacked out are commands that communicate outside the Arena environment and are irreverent to the research.

Table 9: Results from experiments conducted (elements template)

4.3. Phase 2: Elimination of logic blocks

After determining these properties (logic, data or both), the logic building blocks can be eliminated from the experiments. This is because the logic determines the control and direction of entities through the simulation model and does not affect the integration of modules. The logic of the simulation model is within the module and therefore does not effect the inputs and outputs connected to other simulation models. This information is detailed from pages 46 to 47.

4.4. Phase 3: Experimentation

Arena experiments are conducted by investigating a particular Arena building block in a simulation module. The module must have the building block being investigated. Once this module runs successfully as a single module, it can then be duplicated i.e. two modules can then contain the same building block name.

Integration of the two modules takes place depending on the structure of the inputs and outputs of the simulation modules. Since the two modules have the same building blocks, the variable names require changing. This case is the same for other building blocks. Once this has been completed, then other integration issues can be investigated.

This process will be repeated until all the remaining building blocks have been investigated. The issues arising with respect to the integration of modules are derived from the experiments conducted and are explained in chapter five.

The process of elimination of the Arena building blocks began from the common, support, transfer and then the element template. The block template is eliminated, as all block building blocks are logic (phase 2). However, some logic building blocks will be required, as the element building blocks require some of the logic building blocks in order to run.

4.5. Phase 4: Recommendations

A series of recommendations can be obtained from the identified issues. The author believes that these recommendations would be beneficial for the modularity development of simulation models in Arena simulation package. This will be described in chapter five.

4.6. Phase 5: Conclusions

From all the experience gathered and experiments conducted, a discussion and conclusions can be produced to question the feasibility of the integration of simulation models. This discussion will be presented in chapter five.

4.7. An example of experiments conducted

This section gives an insight into the type of experiments conducted. The example below (figure 19) consists of a first module with a conveyor and then the second module that consists of a process and a conveyor. In both these modules, there are also VBA

building blocks. These building blocks are to retrieve external data as well as outputs data to an external source (for e.g. Microsoft® Excel).

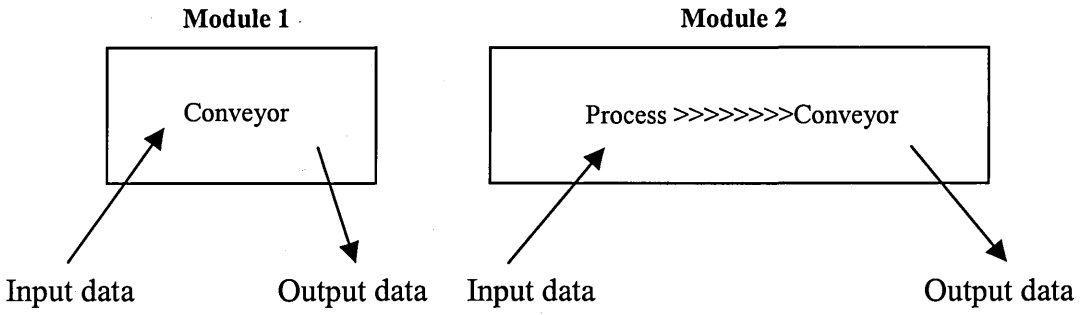


Figure 19: Example of an experiment using VBA

In the simulation software Arena, this is represented by a collection of logic building blocks (figure 20):

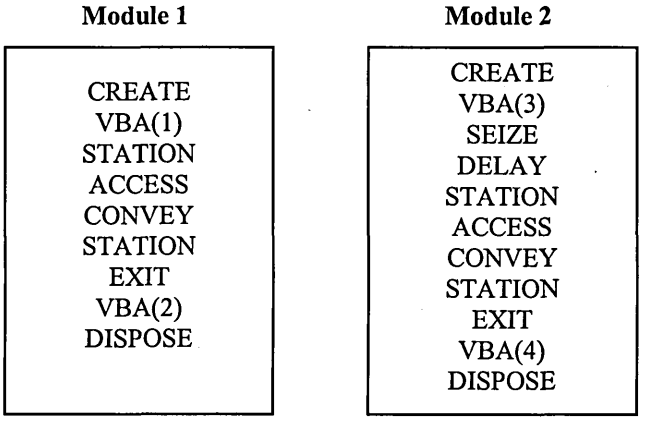


Figure 20: Representation of building blocks in Arena

In order to integrate these two modules together, the DISPOSE building block from the first module and the CREATE building block must be deleted. Thus, permitting the integration of the two modules. This is shown in figure 21:

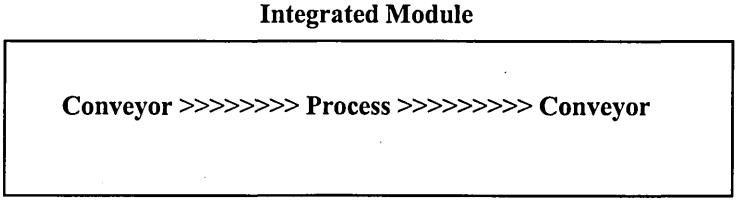


Figure 21: Deletion of DISPOSE and CREATE building blocks for successful module integration

Therefore, the revised Arena building blocks would be (figure 22),

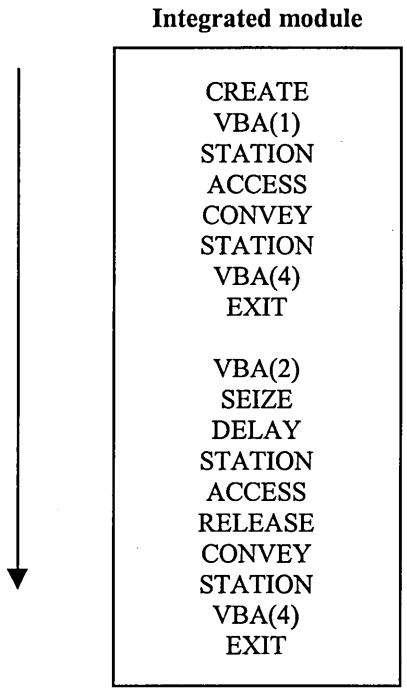


Figure 22: Revised representation of building blocks in Arena

In this example, all conflicts in variable names have been eliminated during the initial experimental stage i.e. using different variable names of the following building blocks:

- STATION
- SEIZE
- ACCESS
- CONVEY
- RELEASE

An example of an integration issue would be the VBA cookie numbers. VBA cookie numbers are internal reference numbers from a VBA building block in the simulation model and this is connected to a particular VBA subroutine code. This can be illustrated in figure 23;

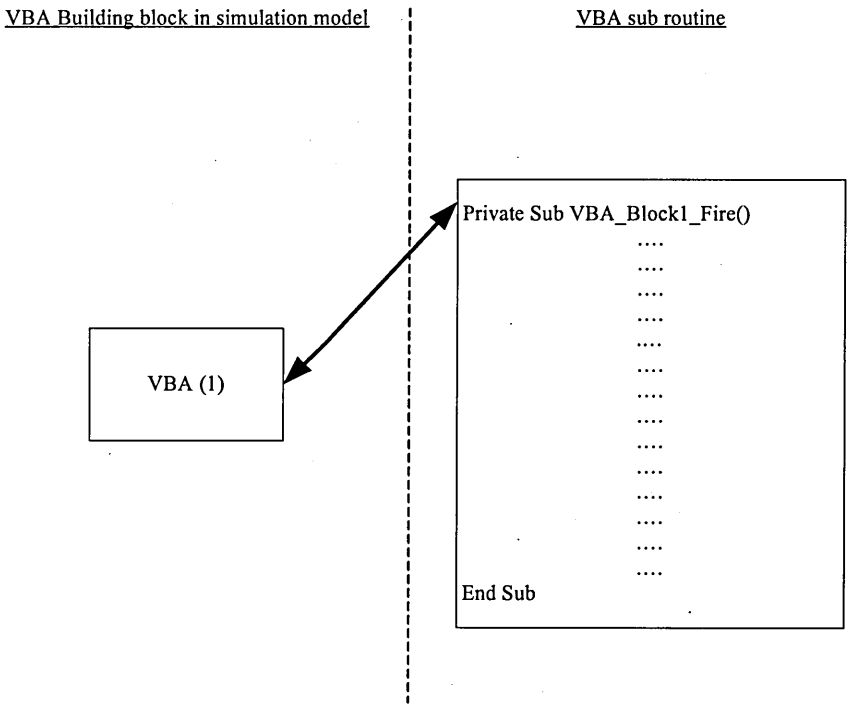


Figure 23: Connection between a VBA building block and a VBA subroutine code

Before integrating the two modules together, the relationship between the VBA cookies and the VBA coding are automatically defined. Once integrated, the VBA cookie numbers lose the relationship. This may not cause an error but may perform an incorrect task (depending on the VBA coding) due to the referencing of these VBA cookie numbers. Therefore, by identifying these issues, the author can establish a list of recommendations in order to answer the hypothesis.

In the next chapter, the results of the remaining issues are presented and discussed.

Chapter 5

Results

5.1. Introduction

In order to investigate the hypothesis (page 31), a number of experiments were conducted. From the methodological process, it is necessary to collect all the data for analysis in order to obtain an answer for the hypothesis. Appendix 1 contains all the results collected and collated. Each identified issue of concern has been allocated with an issue number. These issues surrounding the integration of the modules are reference to a particular building block. The results are split into four sections, common, support, transfer, and elements templates. The results are presented over the next three pages.

5.2. Summary of results

From page 54 to 56 shows a summary of the results from the experiments conducted. Table 10 to 13 shows only the identified issues during the experiment stage. Appendix 1 shows the full results conducted, including the experiments, which contained no integration issues.

5.2.1. Common template results

The table (table 10) below lists a summary of the identified issues from the experiments conducted.

Category	Building Blocks	Integration Issues (Issue number)
Entities and attributes	ARRIVE	Animation properties (1a) Mark Time Attributes (2)
	DEPART	Mark Time Attribute (2)
	SEQUENCES	'Roll on' effect after changing SEQUENCE / STATION name (3)
	SERVER	Animation properties (1b)
Station	ARRIVE	Animation properties (1a)
	ACTIONS	Animation properties (1a)

Resources	ADVANCE SERVER	Animation properties (1b)
	ENTER	Animation properties (1a)
	INSPECT	Animation properties (1b)
	RESOURCE	Utilisation (4)
Queue	QUEUE	Follow on from resource name (5)
	STORAGE	Animation properties (1b)
Transporters	ADVANCE SERVER	Deletion and creation of Network links (6)
Statistics	CONTAINERS	Requires unique number (7)
Run Control	MENU	Only accepts 2nd module MENU building block (8)
	SIMULATE	Last inputted value (9)

Table 10: Summary of results (common template)

5.2.2. Support template results

The table (Table 11) below lists a summary of the identified issues from the experiments conducted.

Category	Building Blocks	Integration Issues (Issue number)
Entities and Attributes	ASSIGN	Lost of WIP (10)
	CREATE	Mark Time Attribute (2)
	READ	When integrating assignments – Lost of assignments which are crucial in defining the variable matrix (10)
Storages	STORE	Animation properties (1b)
	UNSTORE	Animation properties (1b)

Table 11: Summary of results (support template)

5.2.3. Transfer template results

The table (Table 12) below lists a summary of the identified issues from the experiments conducted.

Category	Building Blocks	Integration Issues (Issue number)
Station	ROUTE PATH	Requires unique number (7)
Transporters	FREE	Issue with free to guided and vice versa (11)
	NETWORK LINKS	Deletion and creation of Network links (6)
Conveyors	SEGMENT	Animation properties (1b)

Table 12: Summary of results (transfer templates)

5.2.4. Element template results

The table (Table 13) below lists a summary of the identified issues from the experiments conducted.

Category	Building Blocks	Integration Issues (Issue number)
Station	SEQUENCES	Requires unique number (7)
Queues	Queues	Requires unique number (7)
	Rankings	Queue and ranking names need to change after integration (5)
Transporter	NETWORKS	Requires unique number (7)
Variables	RATES	Requires unique number (7)

Table 13: Summary of results (element template)

5.3. Integration issues

A further detailed examination of the integration issues is explained below using the issue numbers mentioned in section 4.1. Table 14 shows a summary of 'the effects' due to the integration of two or more modules. Theses effects are derived from the experiments conducted.

Issue	The effect
Issue 1a & b	Limited to the use of animation
Issue 2	Deletion of Mark Time Attribute – can be used at a later stage.
Issue 3	Incorrect order of sequence
Issue 4	Validation of integrated simulation model (utilization)
Issue 5	Error with identical queue variable names
Issue 6	Confusion caused by the complexity of network links
Issue 7	Fundamental: Identifying each unique number
Issue 8	Validation errors in multiple use of MENU building block
Issue 9	Validation: Multiple values for the length of replication
Issue 10	Absent of parameters that may be required at a later stage.
Issue 11	Fundamental issue: Transporter

Table 14: Summary of integration issues derived from the experiments conducted

5.3.1. Further detail: Integration issues (Table 14)

Issue 1: Animation properties

In the common template, integration of modules proves to be difficult. There will be inconsistency between the animation properties and logic of Arena commands controlling the animation. Deletion of the ARRIVE building block (Common template) in order to "connect" to other building blocks can cause errors. This is caused by the animation properties which remain active in the system that relate to the Arrive building block, this also applies to the DEPART building block. The DEPART building block also has animation properties. For example, if the module was deleted, the animation properties remain within the system. The animation properties are divided into two categories, 1a and 1b.

Animation: 1A

Each building block is associated with animation properties. These animation properties will be deleted if the building block is deleted. These animation properties will be

associated with other animation properties, thus, causing an error with the simulation model.

Animation: 1B

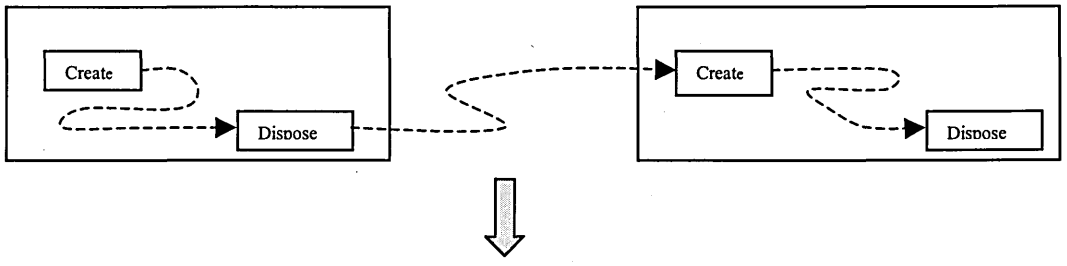
When the user inserts the animation properties associated with the building block, thus, there is a need to be aware of the implications of the animation properties. In comparison, this category is more flexible and gives the user the option of inserting the animation properties than category 1a.

Issue 2: Mark Time Attribute

The mark time attribute plays a crucial information-gathering role by measuring performance, usually by calculating the time of an entity into the system. The attribute parameters are entered in the ARRIVE, CREATE, ARRIVALS building blocks. Thus deleting these building blocks would also delete the mark time attribute, thus rendering the time of an entity invalid.

A possible methodology of verifying and validating each individual module is by calculating the lead-time of a manufacturing system, which can be used as a measurement of performance. In Arena, setting a name in the CREATE module and within the module, the mark time attribute parameter is set. Arena calculates this value by a TALLY building module at the end of the modular system. An issue arises by the integration of two or more modules together. Each module will have its own lead-time attributes and names. When the integration takes place, the CREATE command is removed, but the TALLY building block still remains within the module. Therefore, the TALLY building block must also be removed. Arena also does not allow the same name to be used in all the modules and therefore, the names must to be different to each module.

Before



After

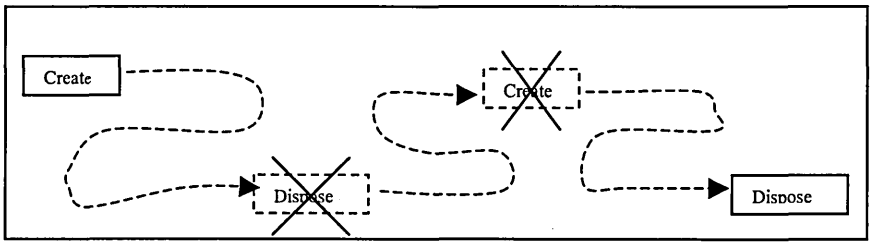


Figure 24 : Showing the integration and deletion of build blocks

As evident from the diagram on the previous page (figure 24), the difficulty arises in removing the creation of entities, upon integration into another module. The entities create a certain pattern (e.g. the time of intervals), which could affect the rest of the modules already integrated together. The entities are created by a user definition; this can be altered when the module is integrated.

Issue 3: Sequences

As shown in the diagram below (figure 25) the integration of two modules can cause a 'roll on' effect. The flow of entities can be altered and the module requires a change in STATION building block names. Sequences will be affected by the name alterations, which in turn change the logic names.

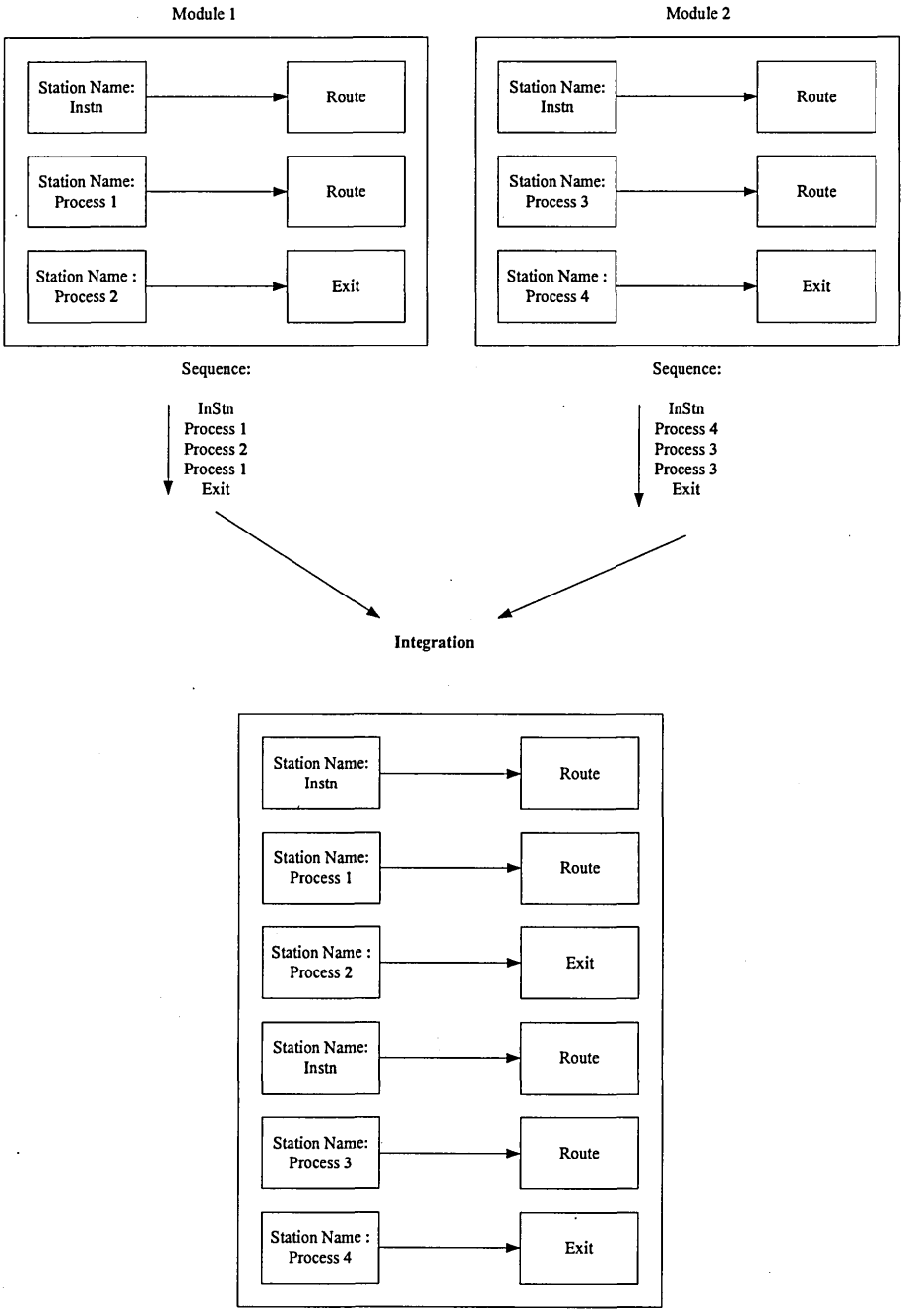


Figure 25 : Alteration of flow of entities entails name changes to the STATION building block

The error occurs when the sequence order could change. This could lead to an incorrect validation of the manufacturing system and therefore, the order of sequences needs to be redefined. An example of the 'roll on' effect could be the change of variable names to adjust for the sequence order.

Issue 4: Utilisation

The Arena software does not report an error with multiple resources, but instead the utilisation of a resource with the same name, which increases the utilisation of one resource. An example of an increase in utilisation is one module with an operator as a resource, where in another module, the resource name is used. When the modules integrate, the software assumes that the resource is the same. But if there were two building blocks with the same resource name, Arena will assume that the resources are sharing the same resource: i.e. the utilisation of the resource will increase. Since, the simulation model is large-scale then the name and number of resources may be over-looked. This will invalidate the results in the simulation model but will give no errors in Arena.

Issue 5: Queues

The Arena software sets individual queue names using the name of the resource, with additional characters (_Q). Therefore, referring to the resource building block can make an assumption; if the resource name is not the same as other resources, then the queues will not affect other queues. An issue arises when two modules have the same queue name. Therefore, only individual modules are valid for integration otherwise an error will occur.

Issue 6: Network links (animation)

For integration of two modules that contain network links, Arena requires them to be deleted before they can be replaced with a different variable name. This can cause confusion if there is a large amount of network links between modules. These network links display a route that the transporter will trek. In order to successfully substitute for a different variable name, network links need to be replaced in the exact place where they were deleted. Thus, this can cause confusion if the user replaces a group of network links and does not know which network link corresponds to which transporter.

Issue 7: Unique numbers

The building block number that relates to SIMAN must be unique, this is important in the low-level template, where they are heavily related to SIMAN. When integrating two modules together, which have two equal building blocks, an error will occur if unique numbers are detected to be identical, thus, names of the building blocks are different.

The user is required to change these numbers if there is a conflict between the modules. Each container used in the Arena model must be unique. Thus, in the SIMAN language, it can be referred to as a unique number. Therefore, integration of these containers can cause a problem, as it is possible to have the same container number, multiple times. This causes an error in the simulation model during a simulation run.

Issue 8: MENU building block

An issue arises when two MENU building blocks are in the same model but the contents are not the same. When each module has a menu system within these modules; furthermore, the integration of these menus can cause a problem.

Issue 9: SIMULATE building block

In Arena, the SIMULATE command (common template), allows two or more of these commands to be in the same active window. However, the simulation software takes the last inputted replication value, and as soon as it is changed, the other simulate modules within each module will change to the last inputted value. This can be useful if each module has a simulate command.

The SIMULATE building block from the common template and the REPLICATE building block (Elements template) are incompatible with each other. Arena will report an error of "invalid element name". If the replicate module was within a module, then Arena will run the simulation model.

Issue 10: Parameters within CREATE building block

Within the CREATE building block, there is an opportunity to set certain parameters, known as assignments. These assignments can be used to define a variable matrix to be used at a later stage in the simulation model. When integrating the two modules together, the CREATE building block is deleted and the assignments within the CREATE building block are also deleted. When these parameters are called upon, this will cause an error within the simulation model.

Issue 11: Transporter

When a transporter (i.e. forklift) is being used in the Arena model and moves from one module to another, Arena does not conflict with the same type. However, an issue arises when a transporter is freely guided in a module, but the same transporter guided (AVG's) is declared in the other module. Arena automatically changes certain parameters. For a module containing a transporter (guided) integrating into the shared module, also containing the same transporter (free), certain parameters change and vice-versa. A diagram (figure 26) below describes the implications with the integration of these modules.

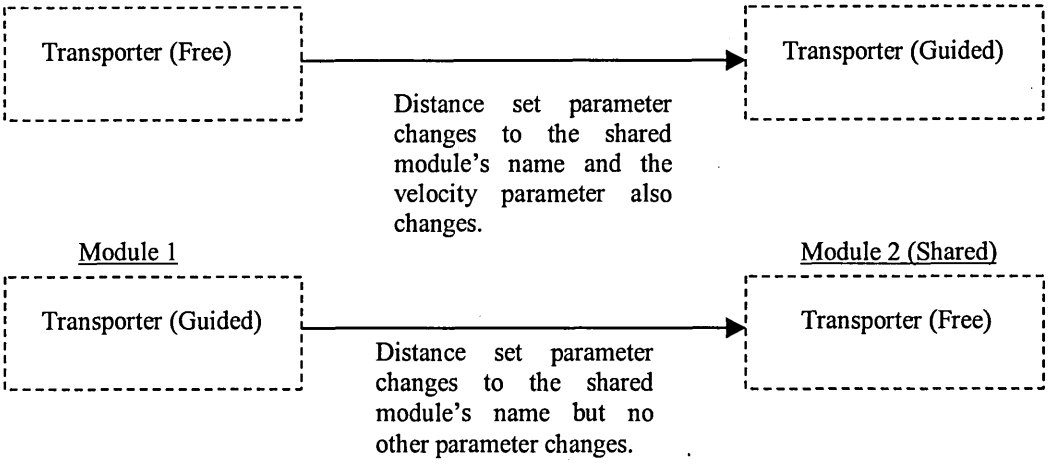


Figure 26 : Implications of the integration of modules

In both cases, Arena does not change the initial position parameters when integrating into the new module. The guided transporter is more complex i.e. more input parameters and less to alter in comparison to the free transporter.

5.3.2. Further detail

From the experiments conducted, table 15 illustrates other issues which is derived from the author's experience during the experimental stage.

Other issues	The effect
Module integration	Inaccurate distributions in the simulation model
Fundamentals of encapsulation	Encapsulation is reduced
Viewable screen	As integrated simulation model increases, the less manageable it becomes.
Visual Basic Application (VBA)	VBA reference cookies automatically change in software package when integrating

File becoming too large	Handling of large simulation model files
Documentation	Confusion of variable names
Verifying connections	Errors with incorrect connections
Excel spreadsheets	In connection with VBA, referencing to Excel sheet can cause errors.
Assumptions	Fundamental: Redefinition of assumptions
Simulate and replicate building blocks	Incompatible building blocks
Redefinition of input variables	Inputs change and need to be refined.

Table 15: Summary of other issues that the author has experienced.

Module Integration

In Arena, validation of modules can be inaccurate due to the distribution created by the CREATE building block. This distribution can be different and depends on the CREATE and DISPOSE building blocks. Assumptions are made with the CREATE module and leading to invalidity of the module integrating into the shared modules, when the CREATE building block is deleted. It therefore integrates into the shared modules. Assumptions would be required for a module to be an exponential that fits for that particular module.

Fundamentals of encapsulation

Encapsulation is defined by Luna (1992) as "..its own variables and implementation details, which are hidden from other modules." In certain commercial software packages, the independent modules are not encapsulated. This provides a one-to-one correspondence in a group of real world objects. An issue arises when the commercial software cannot recognise the difference between the individual modules. Therefore, they are not modules, but one simulation model to the commercial package, with which the internal values and details co-host with other modules.

Viewable screen

As the integration of the simulation models increases, particular modules will be more difficult to be identified. In Arena, the window of the simulation model is limited to the size of the screen as well as the commands surrounding the active window. This can be ambiguous if the simulation model is too large to view.

Visual Basic Application (VBA)

The integration of modules with Visual Basic programming within simulation modules conflicts by the inability to transfer all the VBA block contents into the shared modules:

i.e. the integration of VBA programming. When programming, for example with VBA_Block1_Fire( ), the command codes are integrated into the shared modules. The shared module VBA does not contain the programming for that particular block. Arena also automatically detects VBA blocks when integrating into shared modules and creates new VBA blocks with new references, and consequently ignores the programming codes in the previous block.

A problem arises when the commercial package does not allow the user to reference the VBA unique cookies. A VBA unique cookie is an internal variable used by Arena to reference a particular sub-routine in the VBA coding. This can cause confusion when referencing the VBA logic blocks in the model. An example would be a module (module one) with one to four referenced VBA cookies, doing various functions. Another module (module two) has two VBA cookies with another set of functions. When the two modules integrate together, the newest module integrating into the shared modules, the VBA cookies automatically convert into cookies five and six with no code programming inside these blocks.

File becoming too large

When the integration of the modules are complete, the file that contains all the modular simulations might become too big. The computer running the simulation module may require a higher specification computer capable of running large simulation models. The higher specification consists mainly of the Central Processing Unit (CPU), memory and video card.

Documentation

The lack of accurate detailed documentation could raise an issue concerning the named variables used, and the alteration of variable names. Documentation is required if future modifications are needed for maintenance. Details of the documentation should include the assumptions, variables used and any relevant information made during the development of these simulation models.

Verifying all the connections (individual modules)

It is essential that the simulation model captures the dynamic behavioural characteristics of the system being studied. The quality of the results obtained from the simulation model is only as good as the simulation model being used. Therefore, it is crucial that

the simulation model is as accurate as possible to give accurate results for decision-making. When a module is completely integrated into another module, it requires verification. An issue arises when the module requires to be benchmarked to a real time system or to existing reliable data. If the system does not exist, then the module has nothing to be compared to. It will be assumed that each module will be verified before integration with the rest of the modules.

Excel spreadsheets

In Arena, there is a facility to use data from an external source. Inserting a VBA (Visual Basic® Applications) building block can utilise this data source, which can be in the form of a spreadsheet, e.g. Microsoft® Excel. An issue arises when the shared data are in different formats or linked to different Excel cells. Upon integration, the programming command in VBA is required to be altered. The simulation software has not change. This will cause confusion within the Arena program due to the VBA extracting a particular cell in a spreadsheet if the data in the cell is different.

Assumptions

To give a true representation of a complete system, assumptions must be clearly defined within each module. An example would be a forklift truck. In one module, a forklift could be defined as free transporter building block within the module. The assumption here is that the forklift is defined as a free transporter. In another module, the same forklift could be defined as a guided transporter; i.e. it has a set path for the forklift, but the assumptions are different.

Redefinition of input variables (from individual to the integration of modules)

When all of the modules have integrated, the input values have to be redefined. There may be differences between the integrated modules and individual modules. The initial input values would be different if the initial value came from the output from another module. The diagram below (figure 27) illustrates an example of the redefinition of variables,

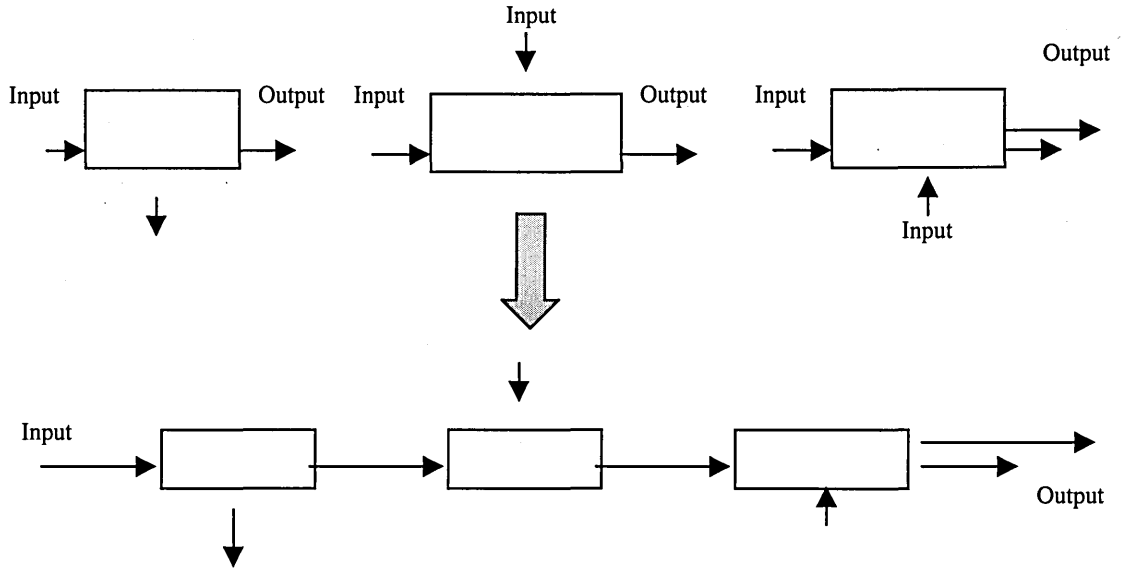


Figure 27: An example of the redefinition of variables

5.3.3. Classification of issues

Errors fall into three types requiring building blocks to be either edited, deleted or created.

Type I: If there is a syntax error within a building block

This category is only required to edit a building block if the name conflicts with another building block name.

Type II: If a building block is deleted

This category will require deleting one or more building blocks in order to integrate two or more building blocks.

Type III: If a building block is deleted and a replacement is required.

This category will require deleting one or more building blocks, it is also required to insert a new building block in order for a successful integration between modules. Table 16 below outlines examples showing each case.

Type I	Type II	Type III
STATION building block	Mark Time Attribute, COUNT, TALLY building blocks	MENU building block
Animation	CREATE AND DISPOSE building blocks	Network Links
Unique numbers	ASSIGN building block	VBA
SIMULATE building block	STATISTICS building block	CREATE building block
Conveyors and SEGMENT building block	CREATE building block	Conveyors

Table 16: Examples of each type

5.3.4. Conclusion

Table 17 shows the frequency of each issue in accordance to which Arena template.

Issue Number	Frequencies	% tage	Common Template	Support Template	Transfer Template	Element Template
1a	3	9.38	3			
1b	7	21.88	4	2	1	
2	3	9.38	2	1		
3	1	3.13	1			
4	1	3.13	1			
5	1	3.13	1			
6	2	6.25	1		1	
7	2	6.25	1		1	
8	6	18.75	1			5
9	1	3.13	1			
10	4	12.50		1		3
11	1	3.13			1	
Total -->	32	100	16	4	4	8

Table 17 - To show the conflicts with the simulation commands

It is evident from the table above, that the common template is the worst affected by conflicting issues. This is mainly due to the animation properties used within the common template. A bar chart, shown on the next page, illustrates the percentages between the four templates (figure 28). The transfer and support templates demonstrates to be the least conflicting, it is also one of the most important issues with reference to the transporter and conveyors. If these are not clearly defined then integration issues could arise.

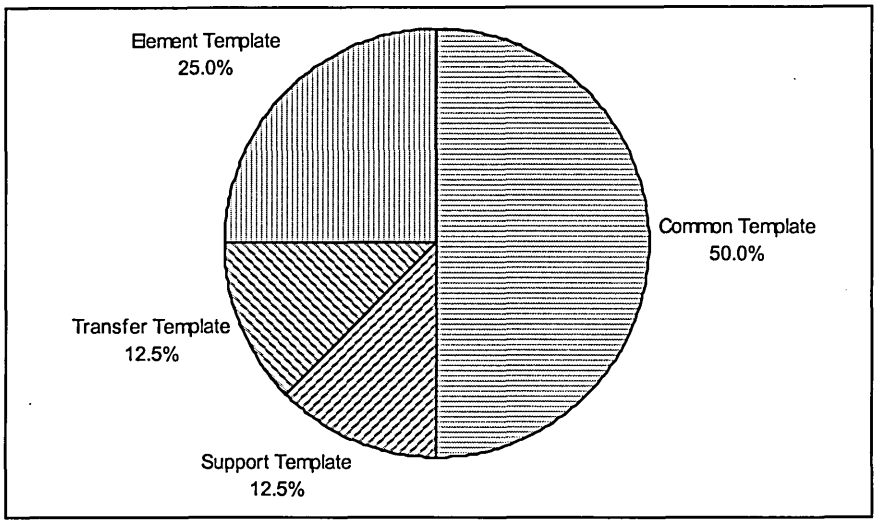


Figure 28 : A pie chart illustrating the proportion of the conflicting issues of four templates that are effected.

The graph above suggests that the Arena building blocks, that should be avoided, are from the common template. This implies that beginners, who normally use the common template, must be aware of, or avoid, the common template, in particular animation properties.

By rearranging the table, the frequencies were put in descending order and tabulated below (table 18).

Issue Number	Frequencies	% tage	Common Template	Support Template	Transfer Template	Element Template
1b	7	21.88	4	2	1	
8	6	18.75	1			5
10	4	12.50		1		3
1a	3	9.38	3			
2	3	9.38	2	1		
6	2	6.25	1		1	
7	2	6.25	1		1	
3	1	3.13	1			
4	1	3.13	1			
5	1	3.13	1			
9	1	3.13	1			
11	1	3.13			1	
Total -->	32	100	16	4	4	8

Table 18: Conflicts with the simulation commands (revised table)

From the graph shown above, Pareto analysis can be applied. Pareto analysis is a technique that highlights the most significant issues involved. By applying the theory, he states that the first 80% are the most significant issues while the remaining 20% are minor issues.

The diagram (figure 29) below shows a graph to showing the percentages of each issue identified.

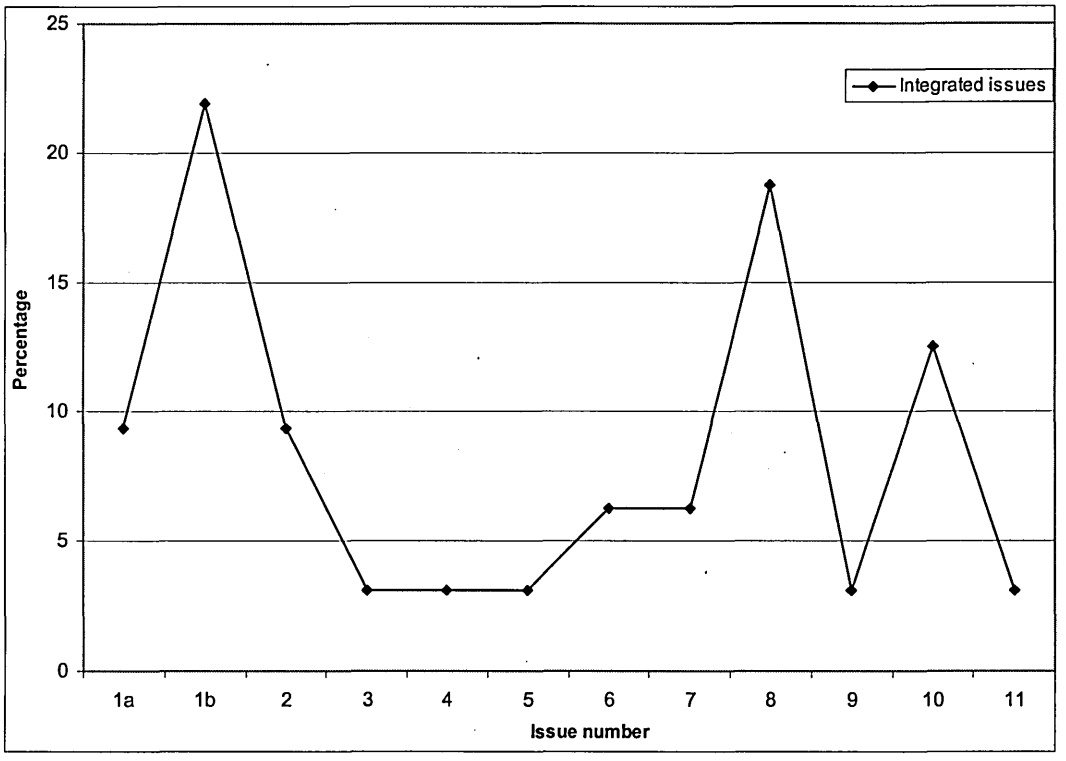


Figure 29 : A graph to show the relationship between the cumulative frequency and the result references

The implications of these results will be discussed in the next chapter.

Chapter 6

Discussion

6.1. Introduction

This section will discuss the experiences encountered with the simulation package during the integration of two modules, and in particular the first seven most significant issues that have arisen according to Pareto analysis (section 4.2.4.). The tables shown below (table 19-20) summarise all the issues derived from the experiments conducted and other issues to be discussed. This chapter also explores the implications of the findings with respect to the industry and the possibility of further research.

Issue number	Description issues
1a & b	Animation
2	Mark time attribute
7	Unique numbers
8	Menu building block
6	Network links (animation)
10	Parameters with CREATE building block

Table 19 : A summary of issues discussed as 'experimental issues'

Other issues
Visual Basic Application
Documentation
Fundamentals of encapsulation
Redefinition of input variables
Distribution entities creation

Table 20 : A summary of issues discussed as 'other issues'

6.2. Experimental issues

6.2.1. Animation

The most frequent problem encountered, as shown in table 18, is the animation properties. They are mostly found in the common template due to the pre-defined construction building blocks with pre-defined animation properties. These pre-defined animation properties restrict the user from integrating 'animation to animation' together.

However, once the integrated modules are up and running, model animation can be a very helpful tool in verifying and validating the integrated simulation models.

6.2.2. Mark time attribute

The second most frequent issue is the mark time attribute, which can easily be removed within the CREATE or ARRIVAL building blocks, where the user is not aware of the deletion. The significance of the mark time attribute within a module can be substantial as it could be used at a later stage in the simulation model. Moreover, when the two or more modules are integrated and the mark time attribute is deleted. The absence of a mark time attribute may present an error. Hence, the author would advise caution when using the mark time attribute as this building block is frequently deleted in order to 'inter-connect' with other modules.

6.2.3. Unique numbers

The author did not experience any difficulty with the unique numbers alteration, particularly in dealing with the elements template. However, it can be confusing if many modules have to be integrated, as building blocks require a unique identification number. This particular issue must be accurately documented, if large amounts of numbers require alteration during the integration process.

6.2.4. Menu building block

From the results in chapter four, it appears that the user must avoid the use of the MENU in independent modules. This may present a problem at a later stage when integrating two or more modules. The author advises the MENU building block to be implemented at the final stage when all the necessary the modules have been integrated.

6.2.5. Network links (animation)

The most difficult issues encountered were the network links. The building blocks are easy to change, however, the entire module must be reviewed, as further changes may be required following the initial building block changes. It is not just the variable name of the network links that require changes but a series of variable name changes within other building blocks associated with the network links. In some cases, it was necessary to change the segment names and other variables that are 'inter-linked' to other network links. This can be confusing and many alterations are necessary, particularly if there are multiple network links.

6.2.6. Parameters within CREATE building block

The integration of these building blocks tends to be at the beginning of each module. Caution is required when deleting the CREATE building block as it is very easy to miss these parameter variables. In most cases, these issues are dealt with at the beginning or at the end of each module. This enhances encapsulation but the network links require more changes, which the user is required to alter within the module by changing the necessary requirements.

6.3. Other issues

Other issues are ranked in order of their complexity beginning with the easiest simulation models to integrate. By applying Pareto analysis to the 'other issues', the first 80% are considered the most significant and are explained below.

6.3.1. Visual Basic Application (VBA)

In manufacturing systems, simulation models are often required to retrieve 'external' data into an Arena simulation model or module. In Arena this is accomplished by using a VBA building block. This can be very confusing if there are many VBA building blocks in one single module referencing to multiple external 'factors' but the VBA cookies automatically change. VBA cookies are internal values used by Arena as a link between the logic (VBA building block) in the Arena model and a particular sub-routine. As a consequence, the user will need to identify and re-correct all the cookie reference numbers. This could prove tedious and time consuming for the user. In particular, users may find it difficult to identify the VBA cookies. This would lead to a loss of valuable time and labour.

An example explained below identifies the above issue during the integration process. The first diagram (figure 30) illustrates a module containing a conveyor and a VBA building block. The representation of this module is a conveyor and the number of entities is counted through the VBA building block. This value is then placed into a particular cell in an Excel spreadsheet, as shown in figure 32. In this example, Arena will communicate with Excel to find the designated cell and place the value in the worksheet. When the integration is complete, all the variable names are unique, however, a problem arises when the simulation package will lose the relationship between the VBA cookies and VBA block. Consequently, Arena will place a designated

value into the wrong cell. This is caused by the loss of relationship between the sub-routine reference number and logic building block number.

Figure 31 illustrates the process by which Arena communicates with an Excel spreadsheet. At the first stage, an entity enters a conveyor that is conveyed to another location, however, in this model the entity leaves the module. Once the entity has left the conveyor, counter mechanism increments by one. Thus, the mechanism keeps track of the total number of entities being conveyed.

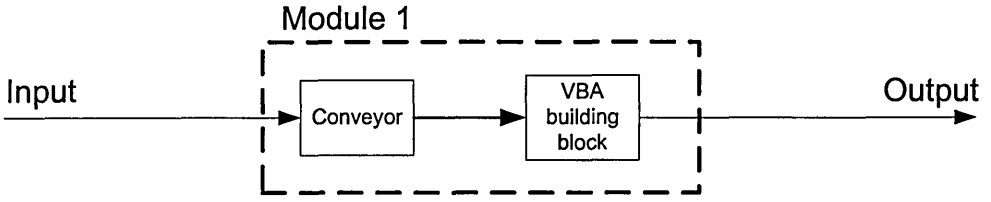


Figure 30 : Diagram showing the representation in a simulation model

The diagram below shows figure 30 being represented in the Arena simulation package.

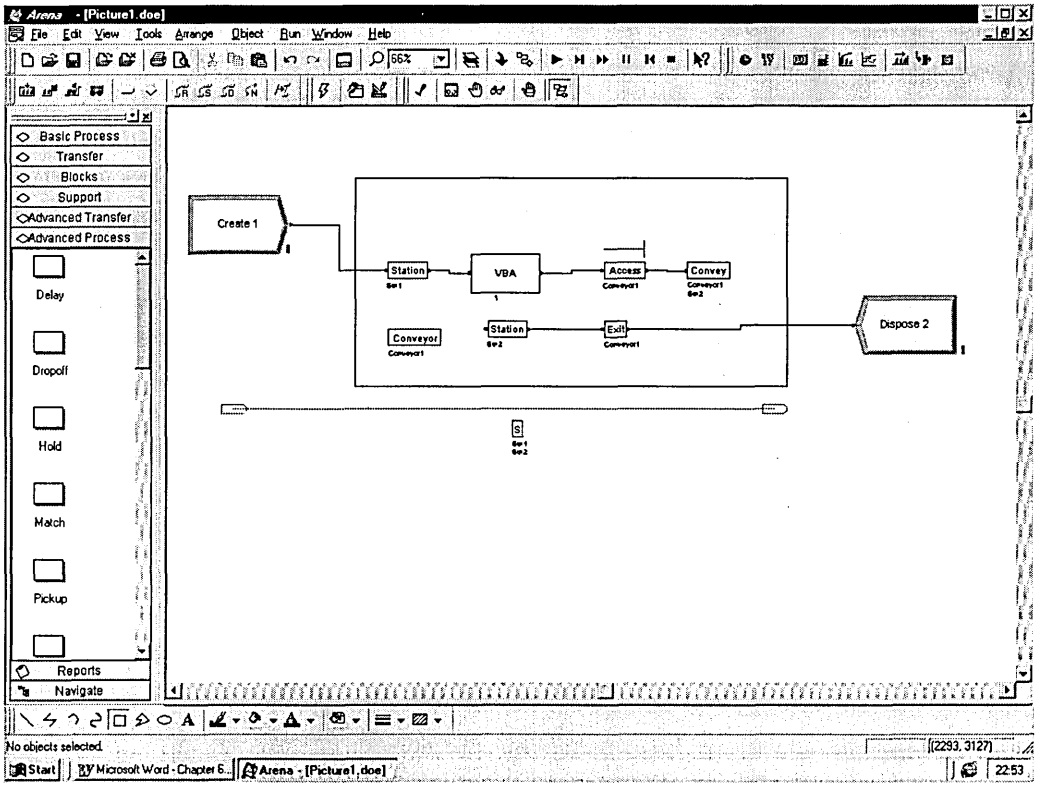


Figure 31 : Representation of the simulation model in Arena

As illustrated in the diagram (figure 32) below communication takes place through a referenced sub-routine and runs a series of commands. The simulation program automatically generates the relationship between the simulation model and sub-routine. These commands open a spreadsheet file and write a current variable value into a designated cell.

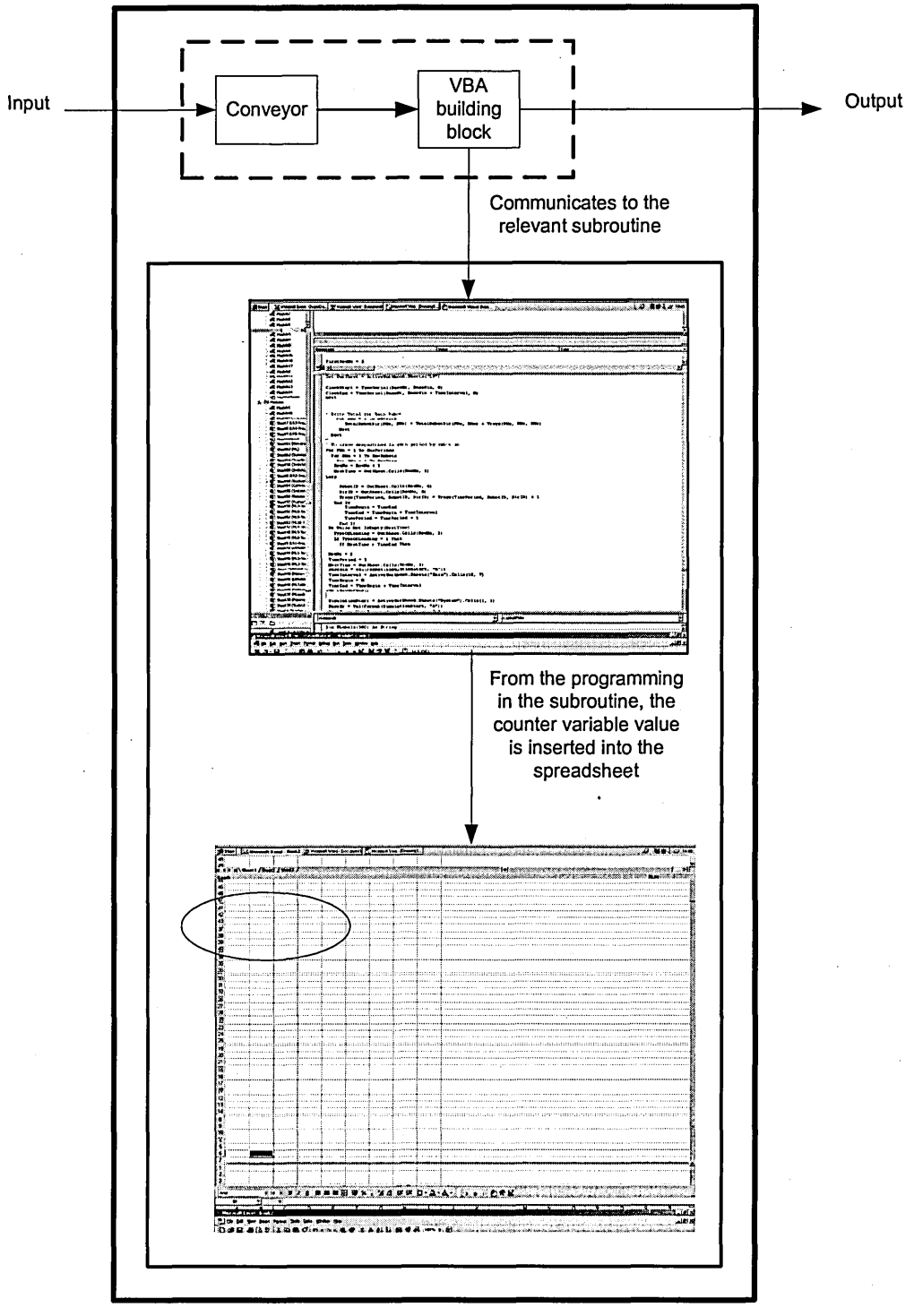


Figure 32: The relationship between VBA model and Excel spreadsheet.

Once the integration between two modules is complete, the relationship between the simulation model and the sub-routine will be lost. Consequently, this could place the designated value to another location, as shown in figure 33.

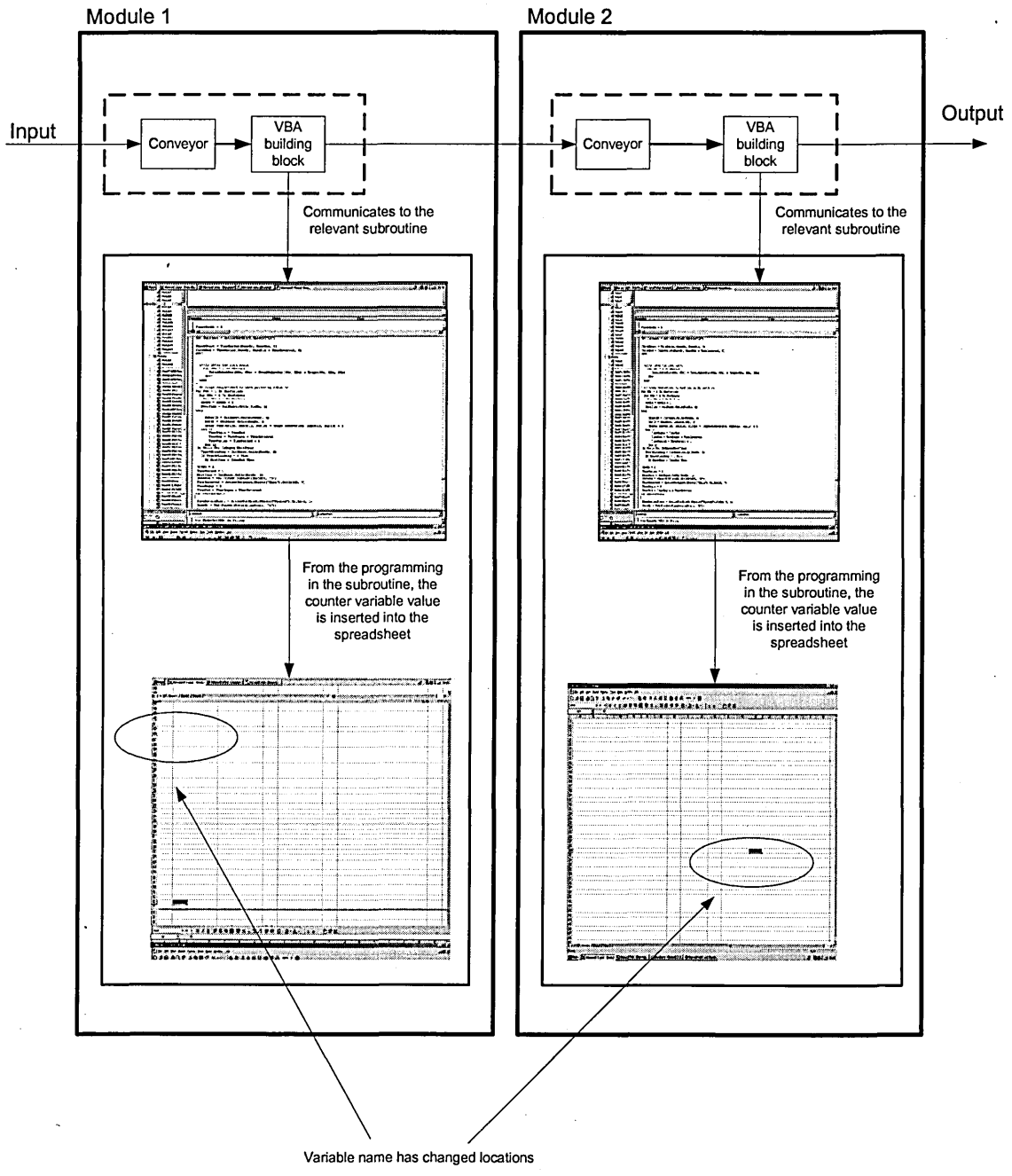


Figure 33: Incorrect VBA referencing during the integration of modules.

It is recommended when integrating two or more modules together that VBA building blocks must be avoided. However, if integration involving the VBA cookies is crucial,

the user must document each relationship between the VBA cookies. The external reference data therefore must be clearly and accurately documented to avoid confusion at a later stage for successful integration.

6.3.2. Documentation

Documentation is essential to facilitate the integration of simulation models. Documentation provides the user with crucial information regarding the variables concerned. Once documented, the user will be able to identify any obvious integration issues.

An example would be the changes required with the VBA cookies to suit the software but without documentation of the relationship between the VBA cookie number and the VBA building block. This would be a very difficult task and time consuming whereby the user must identify the relationships and correct or restructure the VBA cookies.

6.3.3. Fundamentals of encapsulation

This issue is an 'indirect' issue with the integration of simulation models but a fundamental issue with respect to the simulation software. The issue lies with the encapsulation, which requires all the variables to be concealed within a 'black box' or 'module'. This is not the case within the Arena simulation model as shown below; (figure 34)

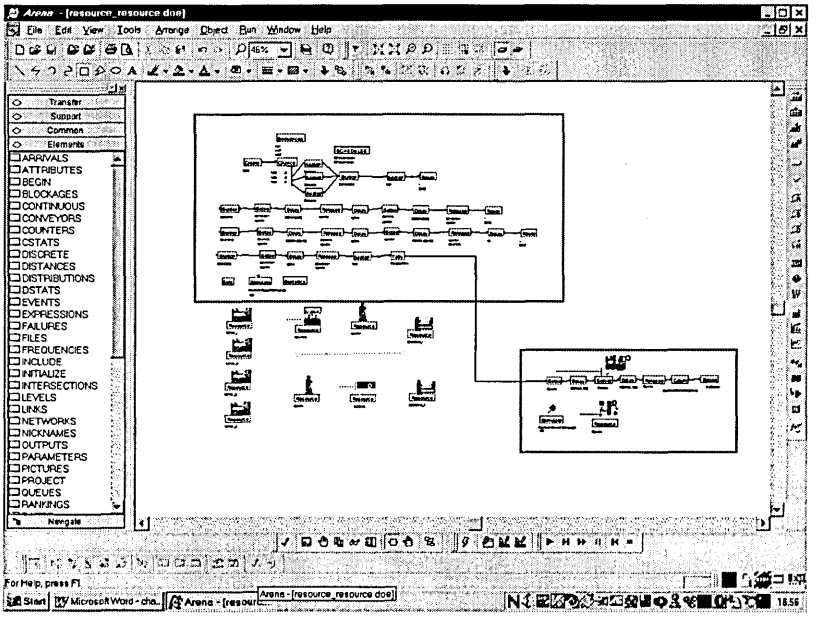


Figure 34 : Diagram showing fundamentals of encapsulation

Arena provides a facility to 'sub-model' and treat the simulation model in a sub-model system as a 'black box'. This facility defeats the objective put forward in the hypothesis of section 2.11 because, as stated in section 1.8, the research focused on one level of hierarchical infrastructure, and when integrating simulation modules; the modules become one large simulation model and lose the identity of independent individual modules. This is the 'downfall' of the simulation software.

6.3.4. Redefinition of input variables

Care must be taken when using independent simulation modules. Since the input data for individual 'modules' has been validated and verified, the input data required needs to be redefined if a new set of data is required for the integrated simulation modules.

6.4. Discussion of the hypothesis

The hypothesis of (2.12) is re-stated.

"Would it be possible to create an integrated simulation model from independent simulation modules simply by changing the variable names?"

This section brings together the results of the research to answer the above question. It is evident from the research and experiments conducted that it is not sufficient to integrate modules, simply by changing variable names.

Following the experiments conducted in chapter four and in the author's opinion a large number of building blocks are required to fulfil the hypothesis. The author used a process of elimination and the errors produced due to the integration of two or more simulation modules are valid. The validation of the results are valid due to the errors occurred as noted in the type I, II and III. Since the simulation package used is very flexible and versatile, it would seem impractical to test every combination that the simulation package has to offer.

The results explained in chapter four, show that there are still errors and fundamental issues that need to be considered. The research has identified the main issues concerned and the approach to the experimentation.

The principle motivation for the use of simulation with commercial packages is to improve efficiency. Simulation must be applied to a situation in order to gain any true benefits. In order to do this, research must be conducted in order to develop the 'grey areas' that are common in modern day industry. The research conducted focuses on the development of a 'grey area' that needs investigation.

The strategy developed by the author is only a tool to assist and minimise the errors that have occurred in the integration of simulation models rather than a solution 'package' that eliminates all errors incurred during the integration process.

On balance for economic and technical reasons, the arguments for the hypothesis outweigh those against.

6.5. Implications for Industry

This outlines the benefits arising from the use of the research recommendations. This can be viewed from three different perspectives:

- Customers and users of a simulation project
- Modellers and analyst providing the service
- Software vendors

6.5.1. Customers and users

Customers and users will not benefit from the recommendations as much as the other perspectives. The main benefits to the customers and users come from the use of smaller simulation models. In particular, the customer may find it easier to visualise a section of the real system than the whole system.

6.5.2. Modellers and analysts

This group of people will benefit most from the recommendations. In particular, using Arena simulation software to construct the simulation models. They will be able to provide a more efficient service to their clients or senior managers. Modellers will be able to construct large-scale models in modular form. Beginners will also benefit, although they will lack experience in the construction of simulation models.

6.5.3. Software vendors

Software vendors can benefit from the research conducted by using the recommendations. In particular, Rockwell software who bought the Arena software rights from Simulation and Modelling Limited. By using this information, later versions could eliminate the issues concerning the integration of simulation models. It is evident from the literature review that a market for this research exists and that the manufacturing industry would benefit greatly from such innovation.

6.6. Limitations

Although the set of recommendations (section 5.7) offer a number of benefits for undertaking the integration of two or more simulation models, it also has a major limitation. This limitation revolves around the fact that the recommendations can only act as a guideline to aid the user for a better integration of modules. Thus, the recommendations are not solutions.

The author's experience suggests that the problems encountered with the integration between two or more simulation models are important issues, which could affect the significance of commercial simulation capabilities.

6.7. Recommendations

From the results in chapter four, a set of recommendations is derived. These recommendations aim to assist the user to successfully integrate two or more simulation modules together. The recommendations are based on the results in chapter four and the experienced gained from the author. Illustrations of these recommendations can be viewed from page 92 to 95 (figures 32, 33, 34, 35). The author emphasises that the recommendations do not provide a solution to all the problems but instead act as a guideline to minimise the occurrence of errors during the integration of two or more modules.

The diagram (figure 35) below shows the overview of the recommendations. The recommendations begin in sequential order with type I issues and then type II, III and finally, other issues.

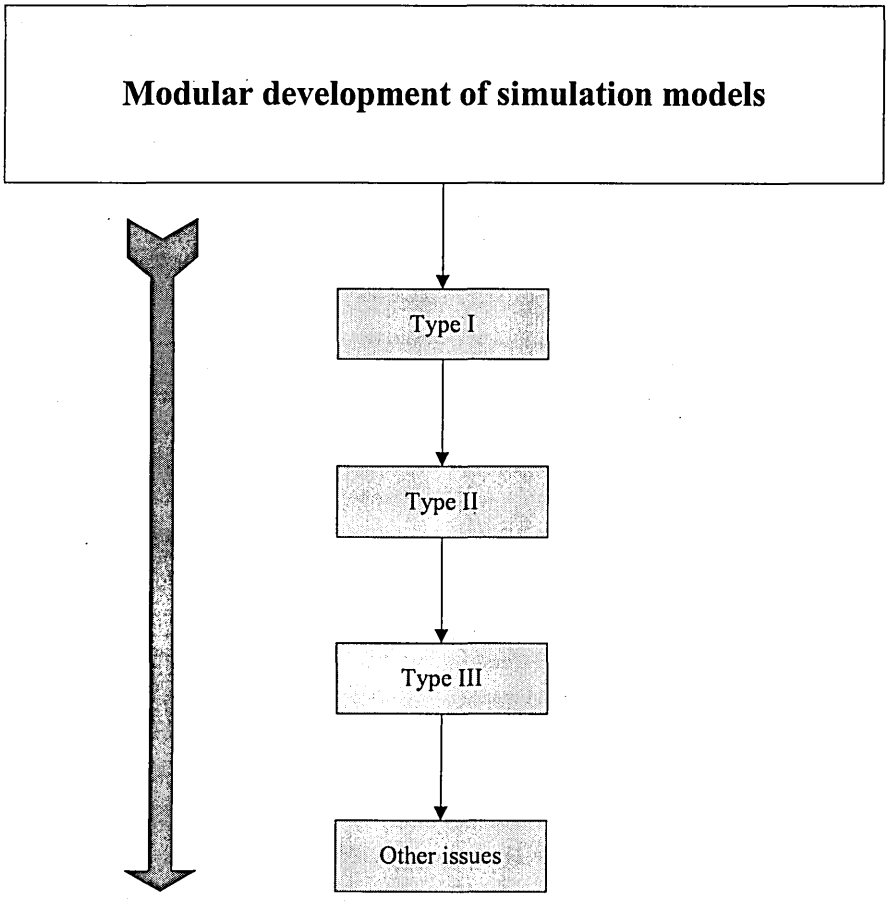


Figure 35 : Overall structure of recommendations

6.7.1. Recommendations for Type I and II

The diagrams (figure 36) below show the recommendations for type I and type II.

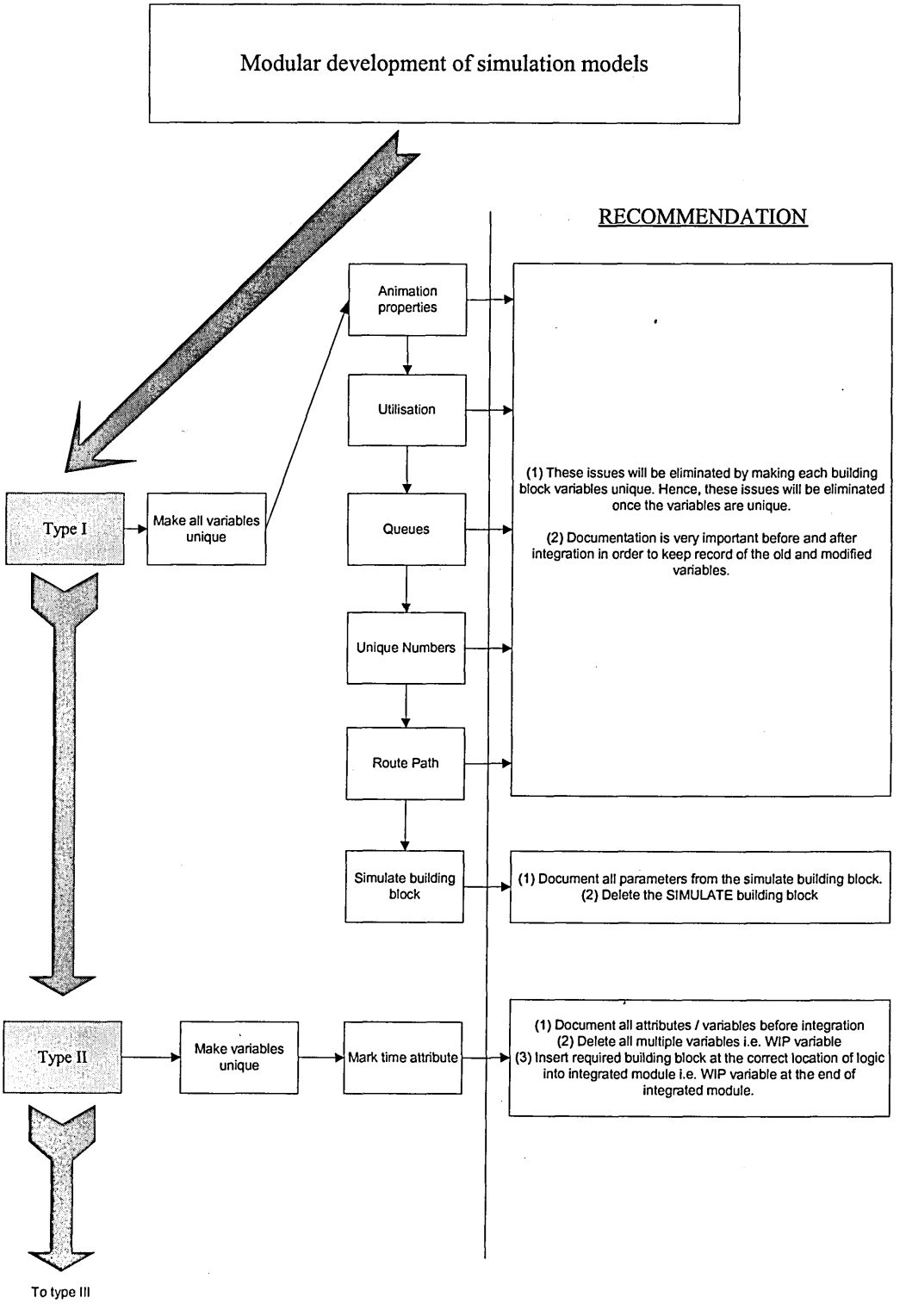


Figure 36 : Recommendations for type I and II

6.7.2. Recommendations for Type III (Continued)

The diagram (figure 37) below shows the recommendations for type II.

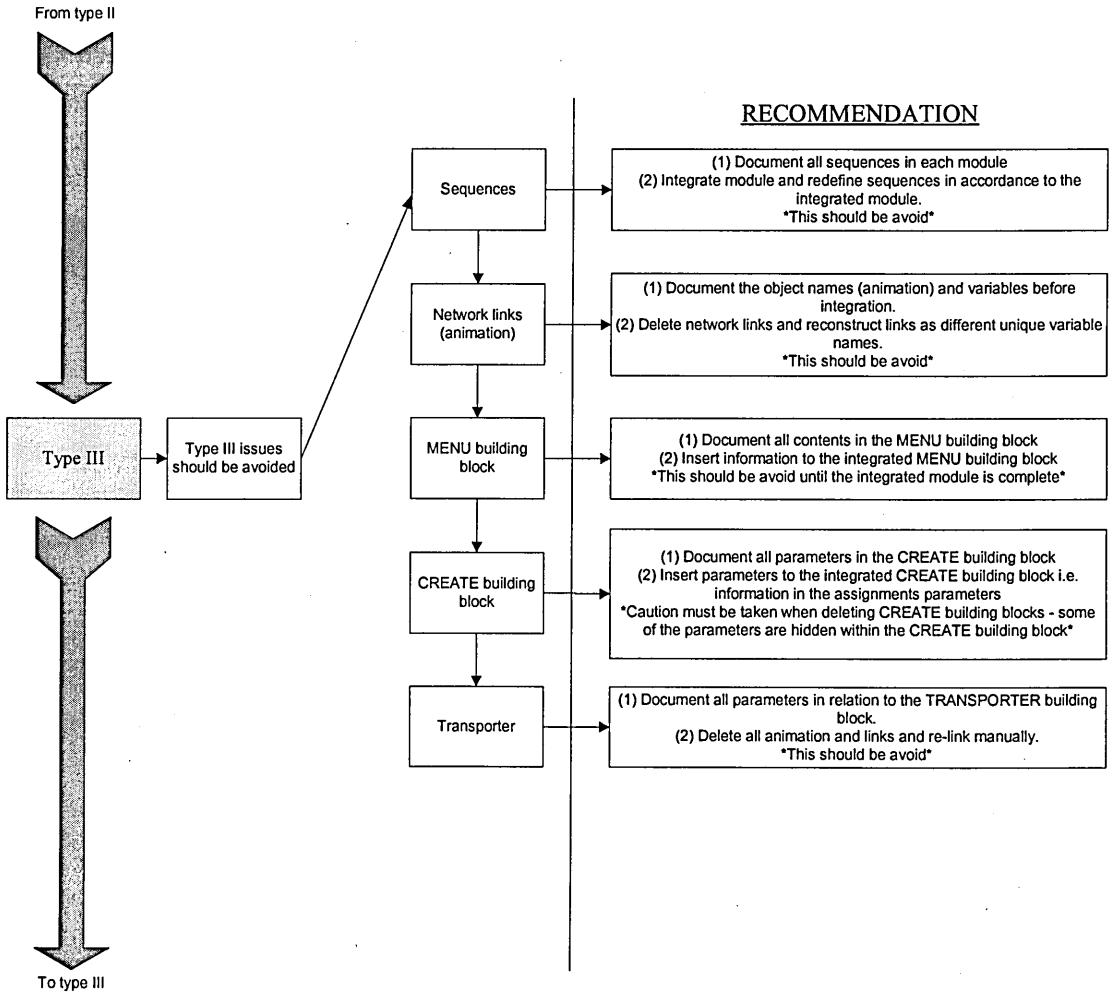


Figure 37 : Recommendations for type III

6.7.3. Recommendations for other issues (Continued)

The diagram (figure 38) below shows the recommendations for other issues.

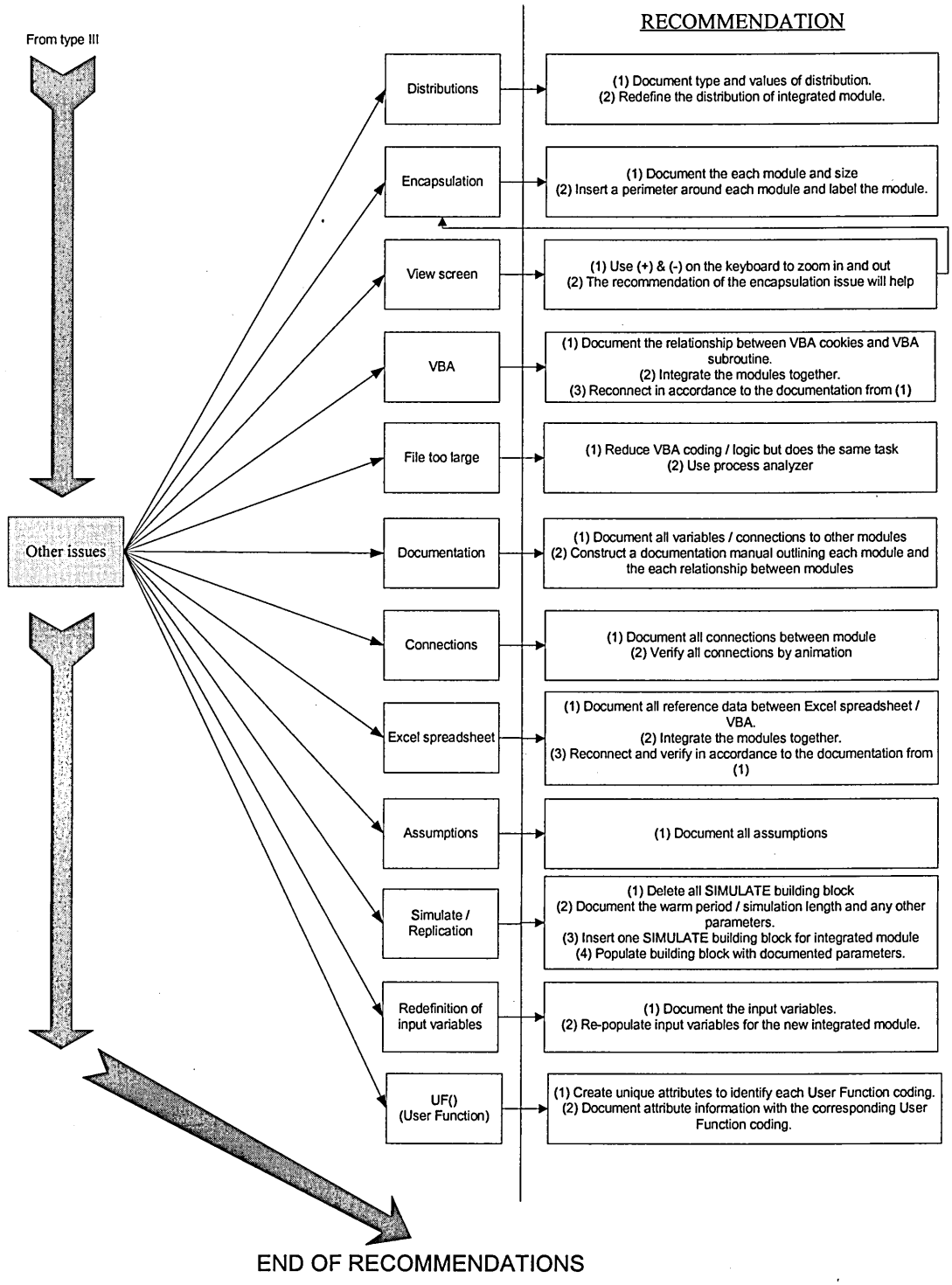


Figure 38 : Recommendations for other issues

6.8. Validation of recommendations

Sweden Post is an international company that use highly automated systems to sort both parcels and letters at their sorting offices in Sweden. Everyday 22 million letters are sorted and distributed by Sweden Post. Sweden Post has 13 sorting centres at different locations in Sweden. Those sorting centres manage both outward and inward sorting. The sorting process, outward, for first class mail is mainly between 17:00 and 22:00. The letters are sorted in different machines and most of the handling with trays is done automatically. The transporting within the sorting centre is handled with conveyors and the loading and unloading with different types of robots.

Since most of the handling with trays is done automatically, an Arena template that is to be built by Sheffield Hallam University should enable Sweden Post to simulate the tray handling process. This will help the decision making process and improve the effectiveness of the conveyor system at the sorting centres.

Sweden Post project provided an excellent opportunity to validate the recommendations as stated in section 6.7. The recommendations (section 6.7.) from the research were applied. The applied recommendations are shown step by step from page 86 to 92.

In the professional version of Arena, there is the possibility of constructing customised modules, which was applied to the Sweden Post project. Each module will have a selection of building task to build and carry out a particular task. In this case, a single module will be constructed in Arena with a selection of building blocks to perform a particular task. The module presents an operator attending to a conveyor and then counts the number of large items being passed the conveyor. This is represented by the following logic in Arena.

Figure 39 shows the building blocks required to accommodate the above situation.

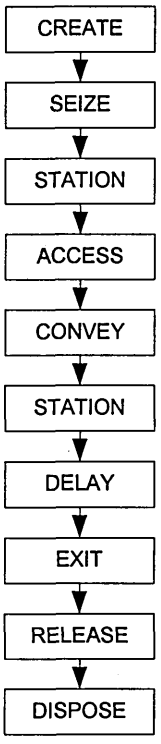


Figure 39: Construction of logic building blocks

The diagram below (figure 40) shows how the logic as described in figure 39 is represented in Arena.

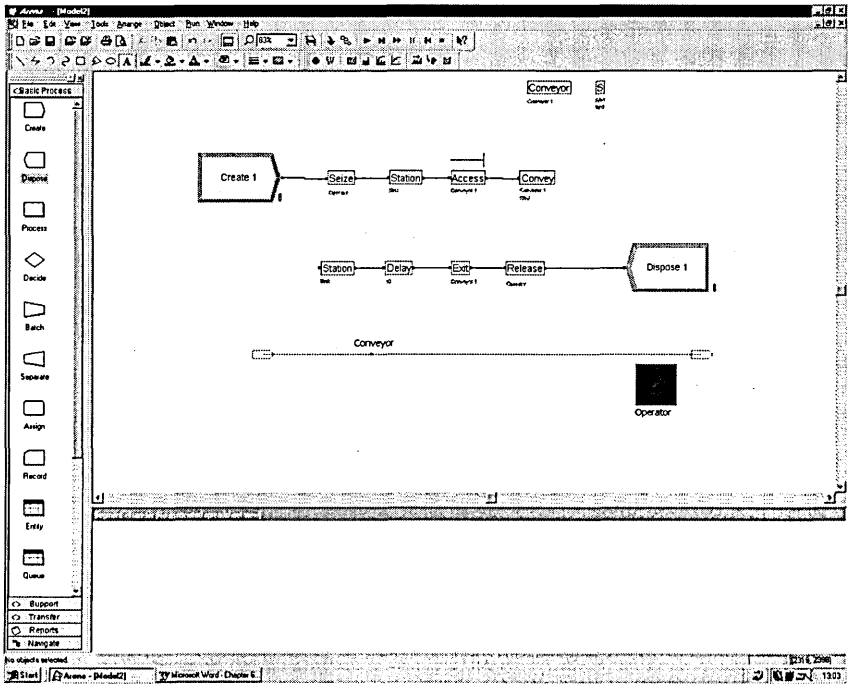


Figure 40: A series of building blocks

The professional version of Arena provides a facility of customising modules to one single building block.

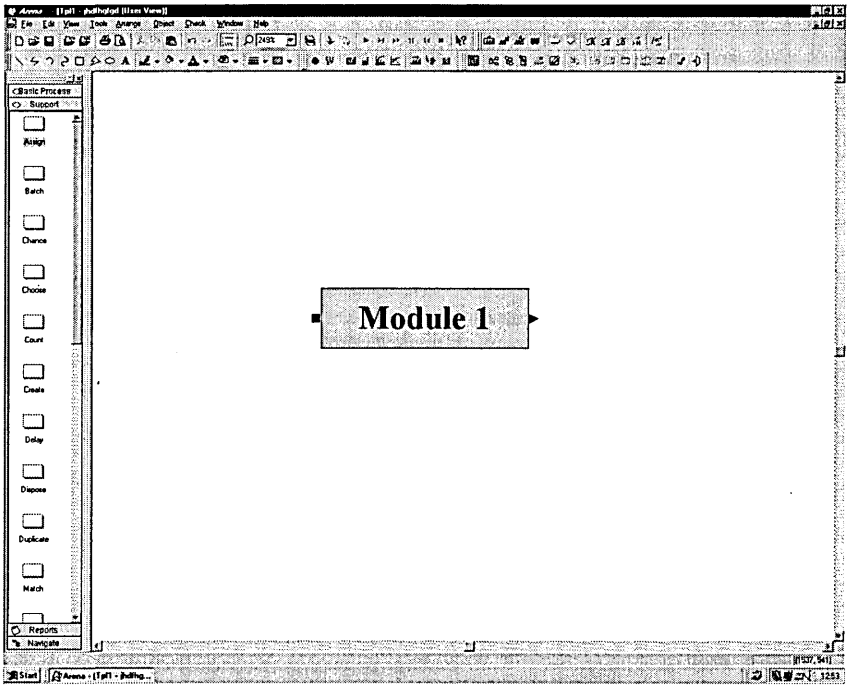


Figure 41: A customised building block

This customised template is then duplicated to produce two customised templates. This is shown figure 42.

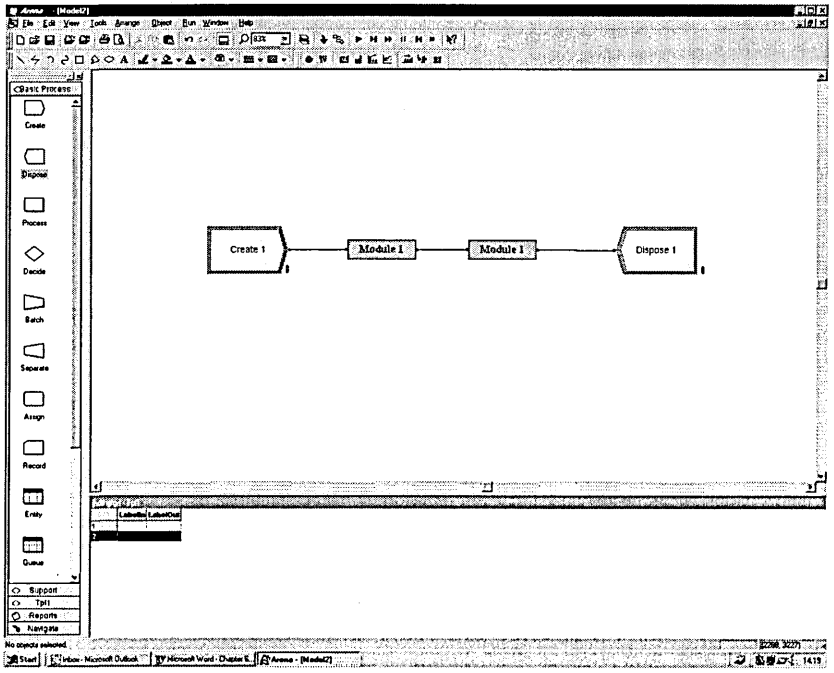


Figure 42: A customised building block

Once the integration was made and the recommendations were applied. The following diagrams show the stages of the validation process. The validation process begins with type I issues. This is shown in figure 43.

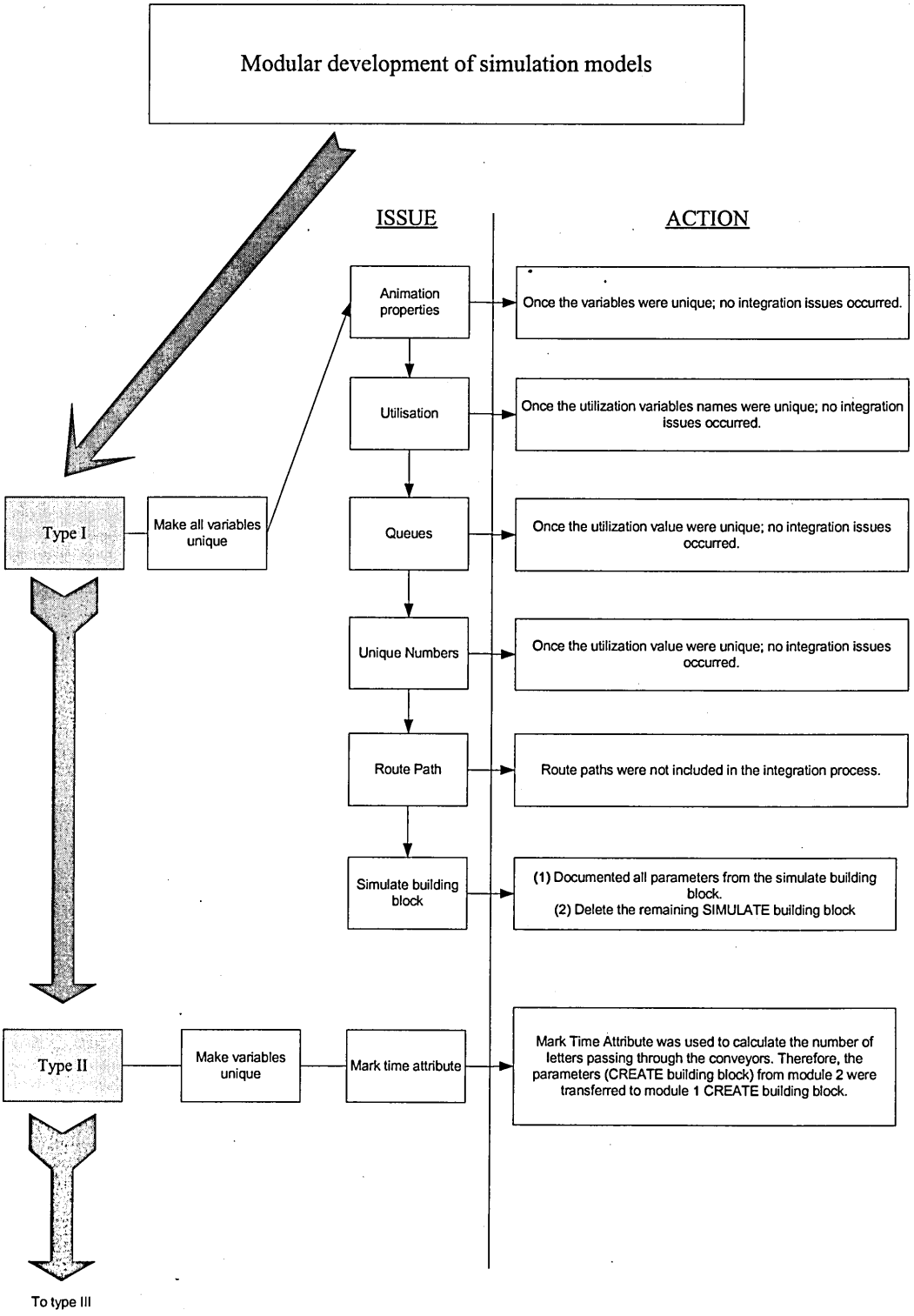


Figure 43: Validation of recommendations for type I and II

Following on from the actions of type I and II, type III of recommendations follows;

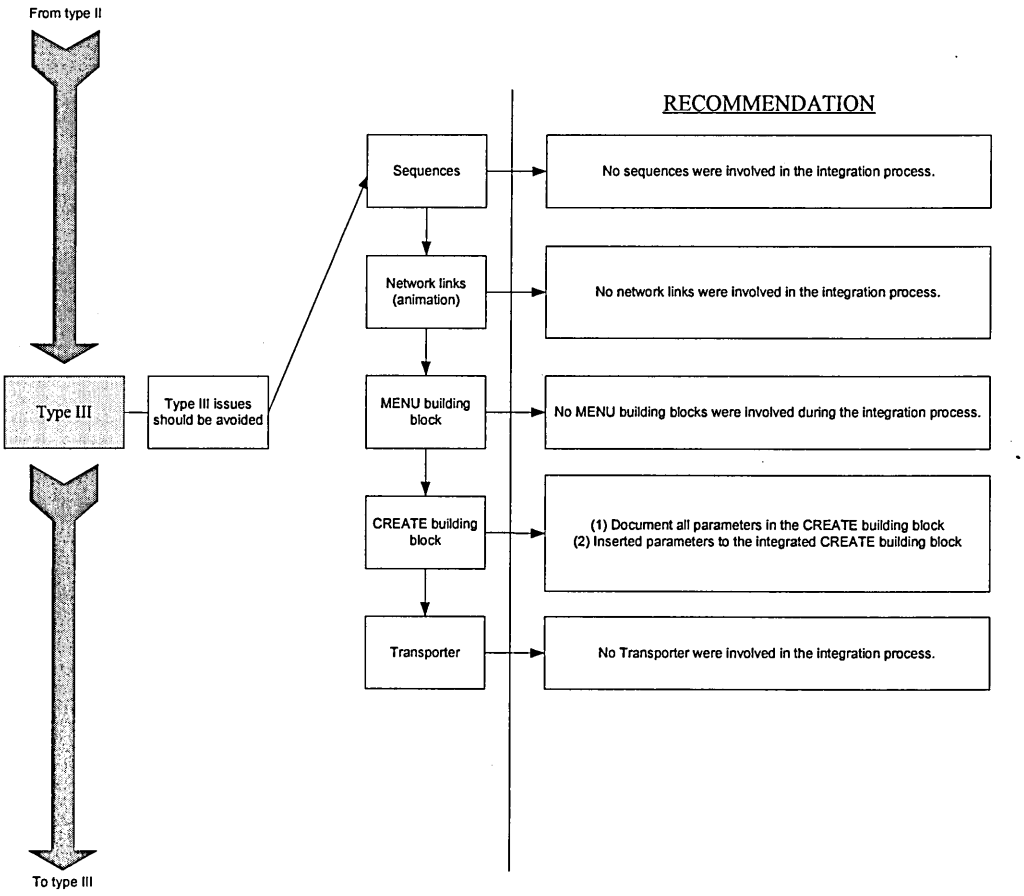


Figure 44: Validation of recommendations for type I and II

Following on from the recommendations of type III, other issues identified from the author are shown below in figure 45.

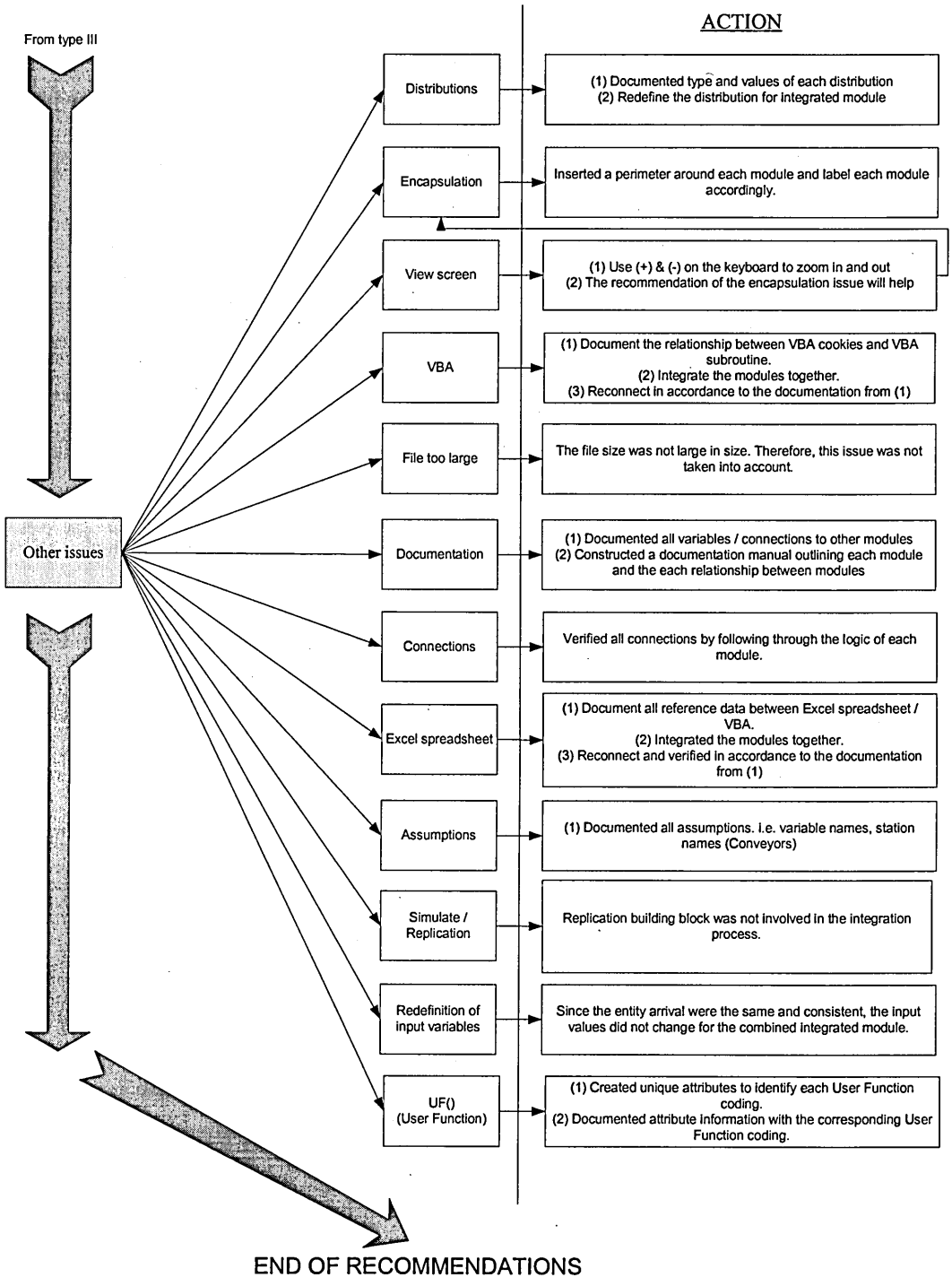


Figure 45: Validation of recommendations for other issues

This concludes the validation process. The simulation modules were on a small scale but if the recommendations were applied to larger scale modules, then the recommendations would be more effective. Thus, there are more interactions within each module.

During the Sweden Post project, a new issue was identified in constructing a new module object. VBA command could not be used, therefore, a User Function, UF(), was used. VBA command could not be used because the VBA reference number could not be altered since the user would not be able to alter the module once complied.

This generic function is identical when integration took place. This problem was solved through the identification of each User Function by a unique attribute attached to the entity to differentiate User Function coding.

An example is given in order to explain how the User Function was applied instead of the VBA building block. In Arena, the User Function programming is generic throughout the simulation model but is required at different parts of the simulation model. In a case scenario, since the programming is generic, programming was needed to cater for all possible situations. The use of attributes was a possible solution if the modules were to be integrated.

The conditions for the attributes were as follows:

Miss process = 1
Carry out process = 2
Delay for 10 seconds = 3

Then the programming structure would be:

```
If Att_Process = 1 Then
 ..
 {Do not perform process}
 ..
Else
```

```
If Att_Process = 2 Then
        ..
        {Perform task}
        ..
        Else
                ..
                {Delay for 10 seconds}
                ..
End If
```

Figure 46 : Programming structure for identifying unique User Function()

The diagram above (figure 46) shows the if the attribute is equal to 1 then a particular task is not performed. If the attribute is equal to 2 then the simulation program will perform a particular task and finally, if the attribute is equal to 3 then the simulation module will stop for 10 seconds before carry on. The programming structure shown above would be common to each module. Therefore, the user would be able to define the outcome without changing the structure of the module or changing the reference numbers if VBA building blocks were used. The new identified issue is now inserted into the recommendations as shown in page 84 (figure 38) as part of the validation process.

The next chapter concludes the final comments to the issues of the modular development of manufacturing simulation models.

Chapter 7

Conclusions

This chapter resolves to provide a definitive conclusion to the issue of the modular development of manufacturing simulation models. In response to increasing demands for more complex simulation models, the technique of modularity appeared to provide the solution to the problem of simulation modelling of large and complex manufacturing systems. Nonetheless, the foundation of the research conducted is based on the hypothesis;

"Would it be possible to create an integrated simulation model from independent simulation modules simply by changing the variable names?"

This chapter will draw on all the evidence obtained from the research process as well as the theoretical discussions undertaken to provide an informed solution to the underpinning hypothesis re-stated above.

The objectives of this research as stated in chapter 1 (page 4 to 5) were met. The first objective is stated below,

Identify the issues involved in the simulation of large-scale manufacturing systems.

Within the thesis, evidence shows the first objective was initially met in chapter 1 where section 1.3 introduced the issues involved in the simulation of large-scale manufacturing systems. Following from section 1.3, the literature review of chapter 2 discusses particular identified issues, including modularity.

The second objective of the research was;

Construct a series of modular simulation models.

The model builder has to undergo a number of highly iterative experiments, which involved the use of the simulation package called Arena in order to meet the second objective. This is evident in chapter 4 where a step by step process of the construction of simulation models is discussed.

The third objective of the research was to;

Study the issues involved with the integration of the modules developed.

The research findings and identified issues during the experiments were discussed in chapter 5 leading to an answer to the hypothesis of the research.

The final objective of the research was to;

Develop a set of recommendations to enable the modular development of manufacturing simulation models.

Derived from the results and from all the points above, a set of recommendations are presented from page 81 to 84. The set of recommendations indicates that simulation practitioners will benefit most from the recommendations and minimise the errors occurred in the integration process.

Further more, evidence shows that contribution to the practice of simulation was met by the following,

Changing the variable name alone is not adequate.

This contribution is evident in chapter 4 where the experiments conducted demonstrate that the user will experience difficulty in changing the variable name during the integration process and required more than just simply changing the name. From the above statement, the research findings was found to be inadequate, thus, sound knowledge or expertise of multiple module integration will be required. The result will contribute to the recommendations and conclusion put forward in this thesis.

The last contribution to the practice of simulation is:

A development of a set of recommendations will minimise the potential problems of integrating modules together.

The set of recommendations is shown on pages 81 to 84. The recommendations provide a guideline for the user to reduce the occurrence of errors, which can also be applied to a real manufacturing scenario. Through evidence, the author found the set of recommendations very useful by minimising the errors occurred through the validation process. The validation process is shown on page 85 to 92.

On the other hand, this chapter also aims to evaluate the research procedure adopted and thus, to suggest possible improvements. In particular, the author's comments and perceptions with respect to the research undertaken in this thesis will also be taken into account.

The author suggests that in order to achieve maximum benefit from the integration of two or more simulation models, the common template should be avoided. Limitation of the construction of simulation models within modules to support, transfer and element templates are advisable. Consequently, it is recommended that beginners, who are new to the simulation package, should not attempt to integrate modules despite the fact that the simulation package is easy to learn. If the integration of these modules is absolutely necessary, then the set of recommendations in chapter five should guide the beginner to reduce the complexity of integrating these modules together. As a final comment, the strategy is a set of recommendations and not a solution to all the integration issues encountered by other users of Arena simulation software.

Research is needed to incorporate some of the best features of object-oriented simulation and distributed simulation with commercial simulation software. For example, model creation based on object oriented simulation principles with a 'built in' simulator will benefit modellers because of the predefined objects, such as operator and machines. From the literature review, it is evident that object oriented simulation is the way forward, but there needs to be a compromise between theory (Research) and applying object oriented simulation in commercial packages (Industry).

The results derived from the experiments indicate that the simulation software still has some fundamental issues remaining that need to be addressed in order to facilitate modularity and object-oriented simulation. There is a gap between commercial packages and object-oriented simulation. It is clear that further research is necessary in order to reduce this gap.

In view of all results and findings from the research process, the author offers the following conclusions in respect of the research described in this thesis.

- The demand for modularity in simulation is a key issue to be considered due to the complexity of manufacturing systems and the need for manufacturing companies to remain competitive. Key authors, in particular, Ziegler (1984, 1987, 1990, 1993), have recognised the need for the continuous development of modularity.

- Problems have been identified regarding the integration of modules. It is apparent that current existing software is not sufficient to handle these problems. As yet there is no facility to simply 'cut and paste' modules.

- Through experiments with the commercial simulation package Arena, it was found that bringing modules together by simply changing names was not an option.

- In consideration of the experiments undertaken, the simulation software still has some fundamental issues remaining that need to be addressed in order to facilitate modularity through object-oriented simulation.

- Encapsulation poses a major problem stemming from the fact that Arena fails to fulfil the precondition of all variables being concealed within a 'black box' or 'module'.

- The main drawback stems from the 'sub-model' facility in Arena that defeats the ultimate objective of integrating individual simulation models. Instead, the 'sub-model' facility produces one large simulation model, which evades the problem stated in the hypothesis.

Nevertheless, the author identifies the limitations of the research embarked upon in this thesis and thus, proposes the following possibilities for consideration when undertaking further research:

- Testing and possibly developing the methodology to include all possible situations that have not yet arisen using Arena software. These would also include commands, which give multiple outputs.

- To transform the recommended methodology to a menu system. This would help the simulation practitioner using the Arena software by providing a problem-solving guide.

Finally, the author stresses that the power of the research and findings offered in this thesis is only confined to Arena. Thus, further research could be conducted on alternative simulation software packages as research with other simulation software might establish a common ground as a basis for further development. It might also suggest alternative ways forward with Arena. The author concludes that the research and experimentation carried out on Arena has produced some valuable results as well as raising interesting issues which are worthy of addressing when undertaking further research. Arena offers great versatility and flexibility however it fails to provide a solution to the key issue of module integration. Thus, giving rise to the need for further research and experimentation with other commercial simulation software packages.

Glossary

Throughout the thesis, terms that are not defined are defined in this section.

Building Blocks: These are the basic components (blocks) for creating simulation models. Each block provides a particular function to carry out a particular task.

Words in Capital Letters: Commands used to describe building blocks used in the Arena® simulation software.

Module: Represents a collection of building blocks with a defined boundary.

VBA, Abbreviation for Visual Basic Application used in the Arena® simulation package.

Data encapsulation: Data encapsulation describes the hiding of data structures and the implementation of procedures called methods to operate on the data of an object. (Narayanon *et al.*, 1998)

Inheritance: Inheritance is a technique for deriving new classes from existing one through creating a subclass. A subclass inherits both data and methods of an exiting superclasses (Narayanon *et al.*, 1998)

Reuse: Reuse is associated with the ability of using the same software elements for several purposes in different applications.

Polymorphism: Polymorphism is the ability to take more than one form. Through polymorphism, the same method results in different behaviour depending on the object to which it is bound. (Narayanon *et al.*, 1998)

References

Arena Internet web page, [Online], last accessed on 9.1.02 at http://www.arenasimulation.com

Arena on-line help, [Online], last accessed on 8.1.02 through Arena simulation software, version 5.0.

Auguston, K., January 1997, Data quality can make or break a simulation, *Modern Materials Handling, pp* 57–59.

Avni, T., September 1999, Simulation modeling primer, *IIE Solutions*, pp 38 – 41.

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

Banks, J. and Gibson, R., November 1998, Simulation Evolution, *IIE solutions*, pp 26–29.

Banks, J., 1999, What does industry need from simulation vendors in Y2K and after? A panel discussion, *Proceedings of the 1999 Winter Simulation Conference,* Phoenix, Arizona, U.S.A., edited by Farrington, P.A., Nembhard, H.B., Sturrock, D.T. and Evans, G.W., pp 1501-1508.

Banks, J., 2000, Simulation in the future, *Proceedings of the 2000 Winter Simulation Conference,* Orlando, Florida, U.S.A., edited by Joines, J.A., Barton, R.R., Kang, K. and Fishwick, P.A., pp 1568–1576.

Banks, J., Carson, J.S. and Nelson, B.L., 1999, Discrete-Event System Simulation, Second Edition, (Prentice-Hall).

Barnes, M.R., 1997, An introduction to Quest, *Proceedings of the 1997 Winter Simulation Conference,* Atlanta, Georgia, U.S.A., edited by Andradottir, S., Healy, K.J., Withers, D.H. and Nelson, B.L., pp 619–623.

Carson, J.S., June 1986, Convincing users of model's validity is challenging aspect of modeler's job, *Computer simulation of manufacturing systems,* pp 74 –86.

Cavitt, D.B., Overatreet, M.C. and Maly, K.J., 1996, A performance analysis model for distributed simulations, *Proceedings of the 1996 Winter Simulation Conference,* Coronado, California, U.S.A., edited by Charnes, J.M., Morrice, D.M., Brunner, D.T. and Swain, J.J., pp 629–635.

Clark, G.M., 1996, Introduction to manufacturing applications, *Proceedings of the 1996 Winter Simulation Conference,* Coronado, California, U.S.A., edited by Charnes, J.M., Morrice, D.M., Brunner, D.T. and Swain, J.J., pp 85–92.

Collins, N. and Watson, C.M., 1993, Introduction to Arena, *Proceedings of the 1993 Winter Simulation Conference,* Los Angeles, California, U.S.A., edited by Evans, G.W., Mollaghasemi, M., Russell, E.C. and Biles, W.E., pp 205 – 212.

Daum, T. and Sargent, R.G., 1999, Scaling, Hierarchical modeling, and reuse in an object-oriented modeling and simulation system, *Proceedings of the 1999 Winter Simulation Conference,* Phoenix, Arizona, U.S.A., edited by Farrington, P.A., Nembhard, H.B., Sturrock, D.T. and Evans, G.W., pp 1470–1477.

Davis, W.J. and Moeller, G.L., 1999, The high level architecture: Is there a better way?, *Proceedings of the 1999 Winter Simulation Conference,* Phoenix, Arizona, U.S.A., edited by Farrington, P.A., Nembhard, H.B., Sturrock, D.T. and Evans, G.W., pp 1595-1601.

Davis, W.J., May 1998, Looking into the future of simulation, *IIE Solutions*, 30, pp 24-30.

Davis, W.J., 1999, Simulation: Technologies in the new millennium, *Proceedings of the 1999 Winter Simulation Conference,* Phoenix, Arizona, U.S.A., edited by Farrington, P.A., Nembhard, H.B., Sturrock, D.T. and Evans, G.W., pp 141–147.

Decker, F., Simulation-Based Operation of Production Plants, [Online], last accessed on 4th November 1999 at http://www.iwb.mw.tumuen chen.de/~dr /esmc94 /esmc 94:Et .html.

Extend Internet web page, [Online], last accessed on 8.1.02 at http://www.imaginethatinc.com

Fujimoto, R.M., 1999, Parallel and distributed simulation, *Proceedings of the 1999 Winter Simulation Conference,* Phoenix, Arizona, U.S.A., edited by Farrington, P.A., Nembhard, H.B., Sturrock, D.T. and Evans, G.W., pp122–131.

Garnett, J., January 1999, The last word on simulation, *IIE solutions,* pp 45- 47.

Harrell, C. and Tumay, K., July 1997, Simulation made easy, *IIE Solutions,* pp 39-41.

Hlupic, V., 1999, Discrete-Event Simulation Software: What the Users Want. *Simulation,* 73, pp 362-370.

Hwang, M.H. and Choi, B.K, 1999, Message–passing architecture and its construction in object-oriented rapid modeling for automated manufacturing system simulation, *Simulation, 72,* pp 90–104.

Joines, J.A. and Roberts, S.D., 1999, Simulation in an object–oriented world. *Proceedings of the 1999 Winter Simulation Conference,* Phoenix, Arizona, U.S.A., edited by Farrington, P.A., Nembhard, H.B., Sturrock, D.T. and Evans, G.W., pp 132–140.

Kamat, V.R. and Martinez, J.C., 2000, 3D visualisation of simulated construction operations, *Proceedings of the 2000 Winter Simulation Conference,* Orlando, Florida,

U.S.A., edited by Joines, J.A., Barton, R.R., Kang, K. and Fishwick, P.A., pp 1933–1937.

Kelton, W.D., Sadowski, R.P. and Sadowski, D.A., 1998, Simulation with Arena, International Edition, (Mc Graw-Hill).

Kochan, A., March 1998, Going all the way with simulation. *Manufacturing Computer Solutions*, pp 47-50.

Krahl, D., 2001, The extend simulation environment, *Proceedings of the 2001 Winter Simulation Conference,* Arlington, Virginia, U.S.A., edited by Peters, B.A., Smith, J.S., Law, A.M. and Kelton, W.D., 1991, Simulation Modelling and Analysis, 2nd edition, (Mc Graw-Hill).

Law, A.M. and McComas, M.G., May 1989, Pitfalls to avoid in the simulation of manufacturing system, *Industrial Engineers*, pp 28–69.

Law, A.M. and McComas, M.G., 1999, Simulation of manufacturing systems, *Proceedings of the 1999 Winter Simulation Conference,* Phoenix, Arizona, U.S.A., edited by Farrington, P.A., Nembhard, H.B., Sturrock, D.T. and Evans, G.W., pp 56 – 59.

Luna, J.J., 1992, Hierarchical, modular concepts applied to an object–oriented simulation model development environment, *Proceedings of the 1992 Winter Simulation Conference*, Arlington, Virginia, U.S.A., edited by Swain, J.J, Goldsman, D., Crain, R.C. and Wilson, J.R., pp.694–699.

Mahajan S. K., Brewer, S.K. and Bien, C.D., 1993, QUEST– Queuing Event Simulation Tool, *Proceedings of the 1993 Winter Simulation Conference,* Los Angeles, California, U.S.A., edited by Charnes, J.M., Morrice, D.M., Brunner, D.T. and Swain, J.J., pp 269–275.

Maxwell, T., 1999, A paris–model approach to modular simulation. *Environmental modelling and software*, 14, pp 511-517.

McLean, C. and Riddick, F., 2001, The IMS mission architecture for distributed manufacturing simulation, USA. [Online]. Last accessed on 8.7.01 at http://patapsco.nist.gov/mel/pubstacking/search.asp

Meinert, T.S., Don Taylor, G. and English, J.R., 1999, A modular simulation approach for automated material handling systems, *Simulation Practice and Theory, 7,* pp 15–30.

Nance, R.E., 2000, Simulation education: Past reflections and future directions. *Proceedings of the 2000 Winter Simulation Conference,* Orlando, Florida, U.S.A., edited by Joines, J.A., Barton, R.R., Kang, K. and Fishwick, P.A., pp 1595 – 1601.

Narayanan, S., Bodner, D.A., Sreekanth, U. and Govindaraj, T., 1998, Research in object-oriented manufacturing simulations: An assessment of the state of the art, 30, pp 795-810.

Nicol, D.M., Balci, B., Page, E.H., Fujimoto, R.M., Fishwick, P.A., Smith, R. and L'Ecuyer, P., 1999, Panel: Strategic directions in simulation research. *Proceedings of the 1999 Winter Simulation Conference,* Phoenix, Arizona, U.S.A., edited by Farrington, P.A., Nembhard, H.B., Sturrock, D.T. and Evans, G.W., pp 1509–1520.

Ninios, P., Vlahos, K. and Bunn, D.W., 1995, Industry simulation: System modelling with an object oriented / DEVS technology, *European journal of operational research,* 81, pp 521–534.

Phillips, T., 1998, Automod by Autosimulations, *Proceedings of the 1998 Winter Simulation Conference,* Phoenix, Arizona, U.S.A., edited by Farrington, P.A., Nembhard, H.B., Sturrock, D.T. and Evans, G.W., pp 213–218.

Pidd, M. and Castro, B.R., 1998, Hierarchical modular modelling in discrete simulation. *Proceedings of the 1998 Winter Simulation Conference,* Washington DC, U.S.A., edited by Medeiros, D.J., Watson, E.F., Carson, J.S. and Manivannan, M.S., pp 383–389.

Praehofer, H., 1996, Object oriented, modular hierarchical simulation modeling: towards reuse of simulation code, *Eurosim,*17, pp 5–8.

Proth, J.M., Wang, L. and Xie, X., 1995, A hierarchical and modular approach to production management based on petri nets: Module simplification, *IEEE,* pp 489–497.

QUEST Internet Homepage. [Online]. Last accessed on 1.7.01 at http://www.delima.com.

Randell, L.G., Holst, L.G. and Bolmsjo, G.S., 1999, Incremental System Development of Large Discrete-Event Simulation Models, *Proceedings of the 1999 Winter Simulation Conference,* Phoenix, Arizona, U.S.A., edited by Farrington, P.A., Nembhard, H.B., Sturrock, D.T. and Evans, G.W., pp 561-568.

Rawles, I., 1998, The Witness toolbox–A tutorial, *Proceedings of the 1998 Winter Simulation Conference,* Washington DC, U.S.A., edited by Medeiros, D.J., Watson, E.F., Carson, J.S. and Manivannan, M.S., pp 223–226.

Roberto, M., March 1997, Breaking out of the box. *Manufacturing systems.* [Online]. Last accessed on 25th November 1999 at http://proquest.umi.com/

Robinson, S., 1994, Successful simulation: Practical approach to simulation projects, First Edition, (McGraw – Hill).

Roborts, R.A. and Dessouky, Y.M, 1998, An overview of object-oriented simulation, *Simulation*, 70, pp 359–367.

Ruiz–Torres, A.J. and Zapata, E., 2000, Simulation based operational analysis of future space transportation systems, *Proceedings of the 2000 Winter Simulation Conference,* Orlando, Florida, U.S.A., edited by Joines, J.A., Barton, R.R., Kang, K. and Fishwick, P.A., pp 1123 – 1131.

Sadowski, D., Bapat, V. and Drake, G., 1998, The Arena product family: Enterprise modeling solutions, *Proceedings of the 1998 Winter Simulation Conference,* Washington DC, U.S.A., edited by Medeiros, D.J., Watson, E.F., Carson, J.S. and Manivannan, M.S., pp 205–212.

Sargent, R.G, 1999, Validation and verification of simulation models, *Proceedings of the 1999 Winter Simulation Conference,* Phoenix, Arizona, U.S.A., edited by Farrington, P.A., Nembhard, H.B., Sturrock, D.T. and Evans, G.W., pp 39–47.

Sargent, R.G., 1993, Hierarchical modeling for discrete event simulation (Panel), *Proceedings of the 1993 Winter Simulation Conference,* Los Angeles, California, U.S.A., edited by Evans, G.W., Mollaghasemi, M., Russell, E.C. and Biles, W.E., pp 569–572.

SIMSOC, [Online], last accessed on 12.2.02 at http://www.mailbase.ac.uk/lists/ simsoc

Simul8 Internet web page, [Online], last accessed on 12.2.02 at http://www.visual8.com

Sisti, A.F., 1998, Large-scale battlefield simulation using a multi–level model integration methodology, [Online], last accessed on 20th September 2000 at http://www.rl.af.mil/tech/papers/ModSim/LgScale.html

Stanley, B., 2001, The Automod product suite tutorial, *Proceedings of the 2001 Winter Simulation Conference,* Arlington, Virginia, U.S.A., edited by Peters, B.A., Smith, J.S., Medeiros, D.J. and Rohrer, M.W., pp 209–216.

Storey, J., 1994, New wave manufacturing strategies, Second Edition, (Paul Chapman Publishing Ltd).

Strandridge, C.R., 1999, A tutorial on simulation in Healthcare: Applications and issues, *Proceedings of the 1999 Winter Simulation Conference,* Phoenix, Arizona, U.S.A., edited by Farrington, P.A., Nembhard, H.B., Sturrock, D.T. and Evans, G.W., pp 49–55.

Swets, R.J. and Drake, G.R., 2001, The Arena product family: Enterprise modeling solutions, *Proceedings of the 2001 Winter Simulation Conference,* Arlington, Virginia, U.S.A., edited by Peters, B.A., Smith, J.S., Medeiros, D.J. and Rohrer, M.W., pp 201–208.

Tye, B., 1999, A study on Model Design in the simulation of Manufacturing Systems. PhD Thesis, Sheffield Hallam University, unpublished.

WebPages High Level Architecture at DoD, USA. [Online]. Last accessed on 1.7.01 at http://www.dmso.mil/index.php?page=64/papers/

Witness Internet web page, [Online], last accessed on 12.2.02 at http://www.Lanner.com

Wong, S.T.W., Mak, K.L. and Lau, H.Y.K., 1999, An object-oriented model for the specification of manufacturing systems, *Computers & Industrial Engineering,* 36, pp 655-671.

Zeigler, B. and Vahie, S., 1993, DEVS formalism and methodology: unity of conception/diversity of application, *Proceedings of the 1993 Winter Simulation Conference,* Los Angeles, California, U.S.A., edited by Evans, G.W., Mollaghasemi, M., Russell, E.C. and Biles, W.E., pp 573–579.

Zeigler, B., 1987, Hierarchical, modular discrete-event modelling in an object-oriented environment, *Simulation, 49,* pp 219–230.

Zeigler, B., 2000, Areas of Research in modeling and simulation, [Online], last accessed at 20th September 2000 at http://www.nationalacademies.org/cpsma /nsb/msh.htm

Zeigler, B.P., 1984, Multifaceted Modeling and Discrete Event Simulation, First Edition, Academic Press, Orlando, Florida, 1984

Zeigler, B.P., 1990, Object–Oriented simulation with hierarchical, Modular models, First Edition, (Academic Press).

Appendix 1:Conducted Arena Experiments

	Cateragory	Building Blocks	Result	Issues with integration of simulation models (Reference Number)
Common Template	Entiites and Attributes	Advance Server	o.k.	
		Arrive		Animation properties (1a) , Mark Time Attribute (2)
		Depart		If delete depart then issues involve COUNT and TALLY with mark attribute also Statistics(2)
		Inspect	o.k.	
		Receipes	o.k.	
		sequences		The roll on effect of changing the sequence name, station etc.(3)
		Server		Animation properties with the same entity picture (1b)
		Statics	o.k.	
	Station	Receipes	o.k.	
		Arrive		Animation properties (1a)
		Sequences	o.k.	
		Actions		Animation properties (1a)
		Leave	o.k.	
		Server	o.k.	
		Inspect	o.k.	
		Advserver	o.k.	
		Sets	o.k.	
	Resources	Advance Server		Animation properties (1b)
		Enter		Animation properties (1a)
		Inspect		Animation properties (1b)
		Process	o.k.	
		Resource		Utilisation (4)
		Server	o.k.	
		Sets	o.k.	
		Statistics	o.k.	
	Queues	Advance Server	o.k.	
		Statistics	o.k.	
		Sets	o.k.	
		Queue	o.k.	Follow on from the resource name (5)
	storages	Sets	o.k.	

	Cateragory	Building Blocks	Result	Issues with integration of simulation models (Reference Number)
Common Template	Sets	Arrive	o.k.	
		Enter	o.k.	
		Advance Server	o.k.	
		Storage		Animation properties (1b)
		Leave	o.k.	
	Transporters	Advance server		Animation properties: Deletion and creation nework Links (6)
		Inspect	o.k.	
		Leave	o.k.	
		Process	o.k.	
		Server	o.k.	
		Statistics	o.k.	
	Conveyors	Advance server	o.k.	
		Inspect	o.k.	
		Leave	o.k.	
		Server	o.k.	
	Sequences	sequences	o.k.	
	Statistics	Depart	o.k.	
		Resource	o.k.	
		Advance server	o.k.	
		Containers		Container no. has to be unique. (7)
		sets	o.k.	
		server	o.k.	
	Run Control	Menu		Only takes 2nd module menu system (8)
		Simulate	o.k.	Last inputted value (9)
	Variables	Expressions	o.k.	
		Variables	o.k.	

	Cateragory	Building Blocks	Result	Issues with integration of simulation models (Reference Number)
Support Template	Entities and Attributes	Assign		Lost of WIP (10)
		Create		Lost of Attributes through deletion of dispose and create during integration (2)
		Delay	o.k.	
		Match	o.k.	
		Read		When integrating assignments they are deleted which are crucial in defining the variable matrix (10)
		Signal	o.k.	
		Tally	o.k.	
		Wait	o.k.	
	Station	PickQueue	o.k.	
		PickStation	o.k.	
	Resources	Station	o.k.	
		Release	o.k.	
		Seize	o.k.	
	Storages	Store		Animation Properties (1b)
		Unstore		Animation Properties (1b)
	Variables	Count	o.k.	

	Cateragory	Building Blocks	Result	Issues with integration of simulation models (Reference Number)
Transfer Template	Station	Route	o.k.	
		Route Path		The name need to be unique and consistent (7)
	Transporters	activate	o.k.	
		Allocate	o.k.	
		Free		Issue with free to guided and vice versa (11)
		Halt	o.k.	
		Move	o.k.	
		Network Link		Need to delete links and insert correct Network Links of the second module (6)
		Request	o.k.	
		Transport	o.k.	
		Transporter	o.k.	
	Conveyors	Access	o.k.	
		Convey	o.k.	
		Conveyor		
		Distance	o.k.	
		Exit	o.k.	
		Segment		Deletion of animation and conveyors (1b)
		Start	o.k.	
		Stop	o.k.	

	Cateragory	Building Blocks	Result	Issues with integration of simulation models (Reference Number)
Element Template	Element and Attributes	Arrivals	o.k.	
		Attributes	o.k.	
		Discrete	o.k.	
		Nicknames	o.k.	
	Station	Sequences		Sequence numbers must be unique (7)
		stations	o.k.	
	Resources	Failure	o.k.	
		Resources	o.k.	
		Schedules	o.k.	
		Statesets	o.k.	
	Queues	Queues		Each queue number has to be unique (7)
		Rankings		The queue and ranking names need to change after integration (5)
	Storage	Storages	o.k.	
	Sets	Sets	o.k.	
	Transporter	Distances	o.k.	
		Intersections	o.k.	
		Links	o.k.	
		Networks		Each network number needs to be unique (7)
		Redirects	o.k.	
		Transporters	o.k.	The transporter number needs to be unique (7)
	Conveyors	Conveyors	o.k.	
		Segments	o.k.	
	Statistics	Counter	o.k.	
		Cstats	o.k.	
		Dstats	o.k.	
		Reports	o.k.	
	Run Control	Output	o.k.	
		Project	o.k.	
		Replicate	o.k.	
		Tasks	o.k.	
		Trace	o.k.	

	Cateragory	Building Blocks	Result	Issues with integration of simulation models (Reference Number)
Element Template	Variables	Begin	o.k.	
		Expressions	o.k.	
		Frequencies	o.k.	
		Initalize	o.k.	
		Levels	o.k.	
		Parameters	o.k.	
		Rates		The number needs to be unique (rate variable number) (7)
		Recipes	o.k.	
		Reportlines	o.k.	
		Seeds	o.k.	
		Statics	o.k.	
		Tables	o.k.	
		Tallies	o.k.	
		Variables	o.k.	