

Sheffield Hallam University

A structured approach to rapid simulation model development.

YAPA, Saman T.W.S.

Available from the Sheffield Hallam University Research Archive (SHURA) at:

<http://shura.shu.ac.uk/20584/>

A Sheffield Hallam University thesis

This thesis is protected by copyright which belongs to the author.

The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the author.

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given.

Please visit <http://shura.shu.ac.uk/20584/> and <http://shura.shu.ac.uk/information.html> for further details about copyright and re-use permissions.

101 835 380 1



Return to Learning Centre of issue
Fines are charged at 50p per hour



REFERENCE

ProQuest Number: 10701231

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



ProQuest 10701231

Published by ProQuest LLC (2017). Copyright of the Dissertation is held by the Author.

All rights reserved.

This work is protected against unauthorized copying under Title 17, United States Code
Microform Edition © ProQuest LLC.

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

A Structured Approach to Rapid Simulation Model Development

S. T. W. S. Yapa

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

March 2006

Abstract

A Structured Approach to Rapid Simulation Model Development

By

S. T. W. S. Yapa

This thesis presents a new methodology to accelerate the simulation model development process. Three research strategies were adopted during the research namely; case study, questionnaire survey and literature survey. Investigations done during the research revealed that among the stages within a simulation project life cycle the model development phase is the longest. Results of the case study research and literature review revealed that two major reasons for the lengthiness are the lack of understanding of the system to be simulated between the modeller and the user, and difficulty in programming. Many researchers have tried to accelerate the process mainly by improving the programming efficiency. However it is important to develop a model which represents the actual system to be simulated. Existing approaches to accelerating the simulation model development process do not guarantee that the model developed in a shorter time is the model which represents the actual physical system. Therefore there is a need for a new methodology to develop a model at a shorter time while ensuring that it represents the actual system.

A simulation model is a piece of software. Therefore, the new methodology was developed by adopting Rapid Application Development approach of software development which emphasises the active user involvement. There are three components of the proposed methodology; Joint Application Development (JAD) team, a CASE tool to develop the simulation software independent conceptual model of the system, and a translator programme to convert the developed conceptual model into the computer simulation model. The proposed methodology accelerates the model development process not only by improving the effectiveness of the process, i.e. development of the model required by the user once but also the efficiency of the process, i.e. development of the model at a shorter time by automating the model development process.

Acknowledgements

First and foremost, I wish to thank my supervisor, Dr. David Clegg. You have been a great mentor and a great friend. You provided not only academic guidance but also the support in solving my personal problems. I would never have finished my studies without your guidance and support. I am proud to have been able to work with you.

Secondly, I am very thankful to Prof. Terance Perara of Sheffield Hallam University and Prof. D. Tantrigoda of University of Sri Jayewardenepura, who provided useful ideas in writing this thesis.

I like to thank all my friends, especially to Sam, Anura Samantilaka and Ruwan, for their friendship and support.

Finally, I would like to extend my deepest gratitude to my wife Manori, and two daughters Poorni and Nipuni, for their unwavering love.

Table of Contents

	Page
Abstract	i
Acknowledgements	ii
Chapter One: Introduction	1
1.1 Background of the Study	1
1.2 Justification for the Research.....	2
1.3 Aims and the Objectives of the Research.....	2
1.4 Outline of the Thesis.....	3
1.5 Summary.....	5
Chapter Two: Literature Review	6
2.1 Simulation.....	6
2.1.1 Types of Simulation.....	7
2.1.2 Systems and Models.....	8
2.2 Stages of a Simulation Project.....	10
2.2.1 Model Building.....	12
2.3 Simulation Software.....	18
2.3.1 Features expecting from a simulation tool.....	21
2.4 Summary.....	23
Chapter Three: Research Design	25
3.1 Research Questions.....	25
3.2 Research Philosophy.....	27
3.3 Research Approach.....	27
3.4 Selection of Research Paradigm.....	29
3.5 Research Strategy.....	30
3.5.1 E-mail Questionnaire Survey.....	30
3.5.2 Case Study Research.....	33
3.5.3 Literature Survey.....	34
3.6 Development of New Rapid Simulation Model Development Framework.....	34
3.7 Summary.....	34
Chapter Four: Case Study Research	36
4.1 Background of the Study.....	36
4.2 Case Study A.....	37
4.3 Case Study B.....	38
4.4 Case Study C.....	40
4.5 Summary of the Discussions.....	42
4.6 Lessons Learned.....	49
4.7 Summary.....	52

Chapter Five: Findings	54
5.1 Reasons for Lengthening Simulation Model Development Process.....	54
5.2 Rapid Simulation Model Development: Current Status.....	58
5.2.1 Model Reuse Approach.....	58
5.2.1.1 Model Reuse: Library Driven Approach.....	59
5.2.1.2 Model Reuse: Library Driven Plus Knowledge Based Approach...	63
5.2.1.3 Model Reuse: Object Oriented (OO) Approach.....	65
5.2.2 Integration with Other Packages.....	67
5.2.3 Knowledge Based / Artificial Intelligence Approaches.....	71
5.3 Review of Existing Rapid Simulation Model Development Approaches	80
5.3.1 Model Reuse.....	80
5.3.2 Library Driven Plus Expert System Approach.....	82
5.3.3 Object Oriented Approach.....	82
5.3.4 Integration with Other Packages.....	83
5.3.5 Knowledge Based/Artificial Intelligence Approach.....	83
5.4 Is there a Need for a New Methodology.....	84
5.5 Summary.....	85
Chapter Six: Results of the Questionnaire Survey.....	87
6.1 Questionnaire.....	87
6.2 Sample.....	89
6.3 Findings.....	89
6.4 Discussion.....	95
6.5 Summary.....	97
Chapter Seven : Software Development Methodologies.....	98
7.1 ‘Waterfall’ or Linear Sequential Model.....	99
7.1.1 Issues in adapting Waterfall Model.....	100
7.2 Rapid Application Development (RAD) Approaches.....	102
7.2.1 Rapid Application Development as Proposed by James Martin.....	104
7.2.2 Dynamic System Development Method (DSDM).....	107
7.2.3 Issues in adapting RAD approach.....	109
7.2.4 Tools under RAD Approaches.....	111
7.2.4.1 Prototyping.....	112
7.2.4.2 Computer Aided Software Engineering (CASE) Tools.....	115
7.3 Summary.....	118
Chapter Eight: Development of the New Methodology.....	119
8.1 Direction for a New Framework for Accelerating Simulation Model Development.....	119
8.2 Use of Software Development Techniques in Simulation Model Development.....	121

8.3	A New Methodology for Accelerating Simulation Model Development. Process.....	123
8.3.1	Joint Application Development Team.....	125
8.3.2	A CASE Tool to Develop Prototypes.....	126
8.3.3	Translation of Conceptual Model to Computer Model.....	130
8.4	How the Proposed Methodology Accelerates Simulation Model Development Process?.....	132
8.5	Validation of the Proposed Methodology.....	136
8.5.1	Validation in Computer Science in General.....	136
8.5.2	Validation of the Proposed Methodology.....	139
8.5.2.1	Case Study.....	140
8.5.2.2	Literature Review.....	142
8.5.2.3	Expert Opinion.....	145
8.6	Summary.....	146
Chapter Nine: Presentation of the Rapid Simulation Model Development Tool.....		148
9.1	Structure of the Tool.....	148
9.2	Initialization of the System.....	149
9.3	Development of the Conceptual Model.....	154
9.4	Construction of the Quantitative Model.....	157
9.5	Presentation of the Model.....	158
9.6	Translation of Conceptual Model into Computer Simulation Model.....	159
9.7	System requirements.....	162
9.8	Summary	163
Chapter Ten: Discussion and Conclusions.....		164
10.1	Conceptual Model Development Tool.....	164
10.1.1	Alternative Approaches to Develop the Conceptual Model.....	164
10.1.2	Purpose of the Tool.....	169
10.1.3	Joint Application Development Approach	173
10.2	Implications to the Simulation Industry.....	175
10.3	Conclusions.....	178
10.4	Contribution to the Knowledge.....	178
10.5	Further Research.....	179
10.6	Summary of the Research.....	182
References.....		185
Appendix I – Questionnaire on Simulation Model development		
Appendix II – Letter to Simulation Experts		
Appendix III– List of Publications		

Chapter One

Introduction

1.1 Background of the Study

Simulation is one of the most powerful tools available to decision-makers responsible for the design and operation of complex processes and systems. The Oxford Advanced Learner's Dictionary presents a definition suitable for any type of simulation, i.e. either physical or mathematical, as the deliberate making of certain conditions that could exist in reality, e.g. in order to study them or learn from them. Computer simulation which is the discipline within which the present research was conducted is defined as "methods for studying a wide variety of models of real world systems by numerical evaluation using software designed to imitate the system's operations or characteristics, often over time" (Kelton et al.,1998). According to Robinson (1994) since the 1950s computer simulation has been used to tackle a range of business problems leading to improvements in efficiency, reduced costs and increased profitability. Simulation has become an indispensable problem solving methodology for engineers, designers and managers in an increasingly competitive world. Therefore, a large number of research studies are conducted by academics and practitioners in the field of simulation. Even though many authors/researchers have divided the simulation process into different stages in different ways, modelling had been identified as a major step in all cases (e.g. Pidd, 1992; Robinson, 1994; Oakshott, 1997; Mehta, 2000; Shannon, 2000; Umeda and Jones, 2001). The excessive time and cost needed for the modelling stage when compared to other stages has made the acceleration of simulation model development a major research topic in the simulation field.

1.2 Justification for the Research¹

The literature suggest that the formulation phase of an abstract model and the construction phase of a computer-programming model for simulation often involve lengthy and costly procedures (So and Lew, 1999; Arons, 1999; Son et al., 2000). Research studies conducted in both Japan and USA have shown that approximately 40% of the total simulation project time is spent on model design (Umeda and Jones, 1997; Trybula, 1994). According to Amico et al. (2000), the length of the development process, which may take years, compounds the problems of developing a useful simulation model. Any methodology leading to the acceleration of the simulation model development process not only saves money and time but also provides more time for experimentation to provide better results for the client. Therefore it is worthwhile to explore the ways of accelerating the simulation model development process.

1.3 Aims and the Objectives of the Research

The aim of this research was to examine and propose a methodology for accelerating the simulation model development process. This led to the formation of the research title

A Structured Approach to Rapid Simulation Model Development.

In finding a new approach the following were set as the objectives of the research

1. Identify the factors leading to lengthening the simulation model development process.
2. Review different rapid simulation model development methods/approaches in order to identify the strengths, weaknesses and limitations of each of the method/approach.

¹ Empirical evidence for justification are found in chapter six

3. Empirically validate the assumptions regarding simulation model development that were made based on the literature review.
4. Construct a new framework for rapid simulation model development based on the results of the above three phases.
5. Demonstrate the feasibility of applying the proposed methodology into a real life simulation project and to draw conclusions.

1.4 Outline of the Thesis

There are eleven chapters in the thesis. Each chapter starts with an introductory paragraph which briefly explains the contents of the chapter and ends with a summary paragraph drawing conclusions from the chapter. The contents of each chapter are outlined below.

Chapter One *Introduction*

This chapter gives an overview of the research and explains the background, justification, aims and objectives of the research as well as outlining the contents of the thesis.

Chapter Two *Literature Review*

Concepts which come within the scope of the research are described in detail in this chapter. Then the simulation project life cycle, simulation model development process and simulation software are discussed in this chapter.

Chapter Three *Research Design*

Research questions which were formulated in achieving the objectives stated in section 1.3 are explained in this chapter. Further, research philosophy, approach, paradigm and strategy used of the present research are also discussed in detail.

Chapter Four *Case Study Research*

Three case studies carried out during the research are explained in detail in this chapter.

Chapter Five *Findings of the Research*

This chapter presents the answers to the research questions which will be formulated in chapter three. This includes a discussion on the current status of the research on rapid simulation model development. This discussion identifies a number of approaches proposed by researchers to accelerate the simulation model development process. Then the strengths and the weaknesses of the each of the identified approach are discussed in detail. Finally, this chapter provides the answer to the question “is there a need for a new methodology for accelerating the simulation model development process?” .

Chapter Six *Questionnaire Survey Results*

Results of an e-mail questionnaire completed by simulation practitioners and academics in order to empirically validate the findings of the literature survey are explained in detail in this chapter.

Chapter Seven *Software Development Methodologies*

Since a simulation model is also a piece of software, various software development approaches and techniques are explained as a prelude to identifying a better approach for simulation model development.

Chapter Eight *The New Methodology for Accelerating the Simulation Model Development process*

The new methodology proposed under the present research is explained in detail in this chapter. This explains the different components of the new methodology and how it

accelerates the simulation model development process. A detailed discussion on the validation of the proposed framework is also included into this chapter.

Chapter Nine *Presentation of the Rapid Simulation Model Development Tool*

This chapter explains the tool (CASCoMoD) developed to assist the modeller and the user to develop the conceptual model of the system to be simulated and subsequently translate the conceptual model into the simulation computer model.

Chapter Ten *Discussion and Conclusions*

Implementation issues of the proposed methodology and the implications for simulation software industry are discussed in this chapter. Latter part of the chapter is reserved to present the conclusions of the research and suggestions for further research on rapid simulation model development.

1.5 Summary

This chapter presented the background of this study, justification for and the objectives of the research. A comprehensive survey of literature on relevant concepts is presented in the second chapter.

Chapter Two

Literature Review

The first chapter explained the background of the study, justification for the research, aims and objectives of the research. This chapter presents the concept of simulation in the context of the present research. Then different stages of a simulation project in general and model building, which is the main focus of the research, in particular are discussed. Finally, the chapter is concluded with a discussion on simulation software.

2.1 Simulation

As mentioned in the first chapter, the Oxford Advanced Learner's Dictionary defines simulation as "the deliberate making of certain conditions that could exist in reality, e.g. in order to study them or learn from them". In fact, "simulation" is an extremely general term since the idea applies across many fields, industries and applications (Kelton et al.,1998). But in the present research, attention was paid on the use of simulation in engineering and business applications such as manufacturing, health care, business process reengineering, supply chain management, banking and other service industries. In this context a number of definitions for simulation can be identified from the literature.

"Simulation refers to a broad collection of methods and applications in mimic the behaviour of real systems, usually on a computer with appropriate software" (Kelton et al.,1998).

"The process of designing a model of a real system and conducting experiments with this model for the purpose of understanding the behaviour of the system and/or evaluating various strategies for the operation of the system" (Shannon, 1998).

“Process for exercising mathematical models through simulated time wherein one or more models can be run with varying values of input parameters to evaluate the effects of interaction among variables” (IMTI,2000).

Oakshott (1997) quotes two definitions for simulation and recommends the use of the latter in business-type applications.

“The modelling of a process or system in such a way that the model mimics the response of the actual system to events that take place over time” (Schriber, 1987).

“The process of designing a model of a real system and conducting experiments with this model for the purpose of understanding the behaviour of the system and/or evaluating various strategies for the operation of the system” (Pegden et al.,1995).

From all these definitions it is clear that simulation is about analyzing the behaviour of a system over the time through imitating that system, by developing a model of the real system under study. Theoretically, simulation can be carried out manually. However, today simulation, generally and within the context of this thesis means computer assisted simulation.

2.1.1 Types of Simulation

There are two main types of simulation namely, discrete event simulation and continues simulation. If the system to be modelled can be represented by a series of discrete events¹,

¹ An event is where the state of the system changes instantaneously.

the method is known as discrete-event simulation (DES). This method assumes that nothing of interest occurs between these events so simulation time can progress from event to event. Systems that can be modelled using the discrete approach are generally systems where queuing mechanisms operate (Oakshott, 1997). Whereas in a continuous model, the state of the system can change continuously over time (Kelton et al., 1998). Examples are flow of a fluid into a tank or the cooling of an ingot after it has been removed from a furnace. It is also possible to have combined discrete-event and continuous simulation in one system.

2.1.2 Systems and Models

Computer simulation deals with models of systems (Kelton et al., 1998). According to Shannon (1998), a *system* is a group or collection of interrelated elements that cooperate to accomplish some stated objectives and a model is a representation of a group of objectives or ideas in some form other than that of the entity itself. Even though the physical models of a system can be built, only the logical (or mathematical models) are considered in simulation. A mathematical model is defined as “a set of mathematical and logical relationships among different elements of the system. In *mathematical* simulation the relations of a system are expressed in mathematical formulae, which can be done in two ways: analytical and numerical. In the case of analytical simulation, the modeller will be able to derive an optimal simulation, i.e. one single solution for the problem. Maximization or minimization, for example cost or distance, problems are examples for this type of problems. Numerical simulation deals with behaviour of systems and not so much with ascertaining optimal solutions. There are two kinds of numerical simulations: Deterministic and Stochastic. Deterministic numerical simulation entails fixing the values of parameters, whereas stochastic numerical simulation uses some kind of distribution function as input

for the variables” (Berends and Romme, 1999) (See Figure 2.1). Simulation is particularly advantageous when the complexity or operational viability of systems under study renders the application of purely analytical models impossible (Silva et al., 2000; Khan, 1999).

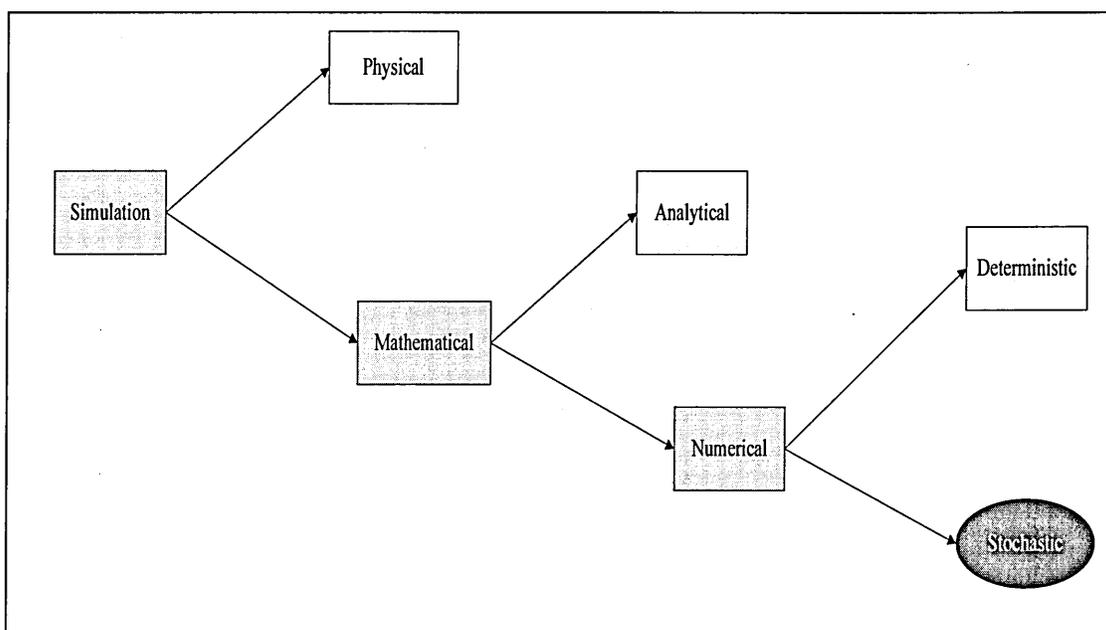


Figure 2.1: Overview of Simulation (Berends and Romme,1999)

Only the systems which can be represented by network of queues are considered in this research. Such a system is modelled by studying the flow of *entities* that move through that system. These entities can have individual characteristics, which are called *attributes*. As the entity flows through the system, it will be processed by a series of *resources*. For example in a factory where parts move from one machine to another, parts are the entities and machines are the resources. Attributes might be things such as name, priority, due date and account number. Most of the parameters related to such a system such as arrival rate of parts and processing time in a machine are probabilistic. However, when the probabilities of parameters become either 0 or 1 depending on the situation, a stochastic system becomes a deterministic system. Therefore, deterministic simulation is a sub set of stochastic

simulation. However, in real life, most of the systems belonged to the above mentioned category have probabilistic parameters. Hence, the research is mainly focussed on stochastic simulation.

In order to analyze the behaviour of a system, it is necessary to build a simulation model and to conduct experiments by using the built model. It is a lengthy process consisting of several stages. These stages are explained in detail in the next section.

2.2 Stages of a Simulation Project

According to Banks (2000(a)), simulation is used to describe and analyze the behaviour of a system and, ask “what if” questions about the real system, and aid in the design of real systems. Both existing (as-is) and conceptual systems (to-be) can be modelled with simulation. According to Banks (2000(b)), simulation is not only a software. It is a multi disciplinary technique that requires a fair amount of training, skills and experience to perform effectively. Simulation is a multi-stage process. Many authors have divided this process into different stages in different ways (see table 2.1). Some argue that each stage must be carried out in a satisfactory manner before the next stage is started (Oakshott,1997; Shannon,1998; Mehta,2000; Banks,2000; Trybula,1994) whilst others argue that simulation is an iterative process (Robinson,1994; Tye,1999). For example, as suggested by Robinson (1994), experimentation may identify some additional issues which alter the definition of the problem and require further model building before experimentation continues. According to Son et al. (2000), the degree of difficulty in building models, the fidelity of the visualization, and the sophistication of the analysis tools vary dramatically. Consequently, building, running and analyzing a simulation model can be a time-consuming and error-prone process.

<ul style="list-style-type: none"> i. Problem definition ii. Project planning iii. System definition iv. Conceptual model Formulation v. Preliminary Experimental design vi. Input data preparation vii. Model translation viii. Verification and validation ix. Final experimental design x. Experimentation xi. Analysis and interpretation xii. Implementation and documentation <p>(Shannon, 1998)</p>	<ul style="list-style-type: none"> i. Project definition <ul style="list-style-type: none"> - Establish objectives - Scope and level of detail - Data collection ii. Model building and testing <ul style="list-style-type: none"> - Structure model - Build model - Verify model - Validate model iii. Experimentation iv. Project Completion <ul style="list-style-type: none"> - Documentation - Present results - Implementation <p>(Mehta, 2000)</p>	<ul style="list-style-type: none"> i. Problem formulation ii. Setting objectives and overall project plan iii. Model conceptualization iv. Data collection v. Model translation vi. Validation vii. Verification viii. Experiment design ix. Production runs and analysis x. More runs xi. Documentation and Reporting xii. Implementation <p>(Banks, 2000)</p>
<ul style="list-style-type: none"> i. Problem Definition ii. Problem Analysis iii. Data Gathering and Validation iv. Model Development v. Model Verification and Validation vi. Model Experiments vii. Analysis of Results viii. Conclusion and Recommendations <p><i>Note : Above stages are normally done consecutively not in parallel.</i></p> <p>(Trybula, 1994)</p>	<ul style="list-style-type: none"> i. Data collection/ gathering ii. Model design iii. Animation iv. Model modification v. Simulation experiments vi. Summary of result <p>(Umeda and Jones, 2001)</p>	<ul style="list-style-type: none"> i. Formulate the problem and plan the study ii. Collect and analyse the data iii. Build the conceptual model iv. Check the validity of the conceptual model v. Develop the computer model vi. Verify (or debug) the computer model vii. Validate the model <p>(Oakshott, 1997)</p>
<ul style="list-style-type: none"> i. Modelling ii. Computing iii. Experimentation <p>(Pidd, 1992)</p>	<ul style="list-style-type: none"> i. Specification ii. Design and development iii. Experimentation iv. Implementation <p><i>Note: This is a highly iterative process.</i></p> <p>(Tye, 1999)</p>	<ul style="list-style-type: none"> i. Problem definition ii. Model building and testing iii. Experimentation iv. Project completion and implementation <p>(Robinson, 1994)</p>

Table 2.1: Stages of a Simulation project – Different Views

Three major generic steps in a simulation project development process can be identified among the divisions given in table 2.1; i.e. **identification of the problem/ problem formulation , model building, and experimentation and implementation.** Model building phase, which is the phase under study in the present research, of the simulation project life cycle is discussed in detail in the next section.

2.2.1 Model Building

Simulation modelling is a practical and yet intellectually challenging activity, one which benefits from careful thought and planning. As pointed out by Pidd (1996) models should be developed gradually. If circumstances permit, the best approach is to develop a simple model and then to collect data to parameterise and test it. It may then be clear that the simple model is fine for the intended purpose or it may be that the model needs to be refined.

i. Structure the model ii. Build the model - Coding - Documenting - Verifying iii. Validate the model (Robinson,1994)	i. Structure Model ii. Build Model iii. Verify Model iv. Validate Model (Mehta,2000)	i. Conceptual Model Formulation ii. Model Translation iii. Verification and Validation (Shannon,1998)
i. Model Conceptualization ii. Model Translation iii. Verification iv. Validation (Banks,2000)	i. Build the conceptual model ii. Check the validity of the conceptual model iii. Develop the computer model iv. Verify (or debug) the computer model v. Validate the model (Oakshott,1997)	i. Model Development ii. Model verification and validation (Trybula,1994)
i. Model Formulation ii. Model Development (Tye,1999)		

Table 2.2: Stages of the Model Building Process

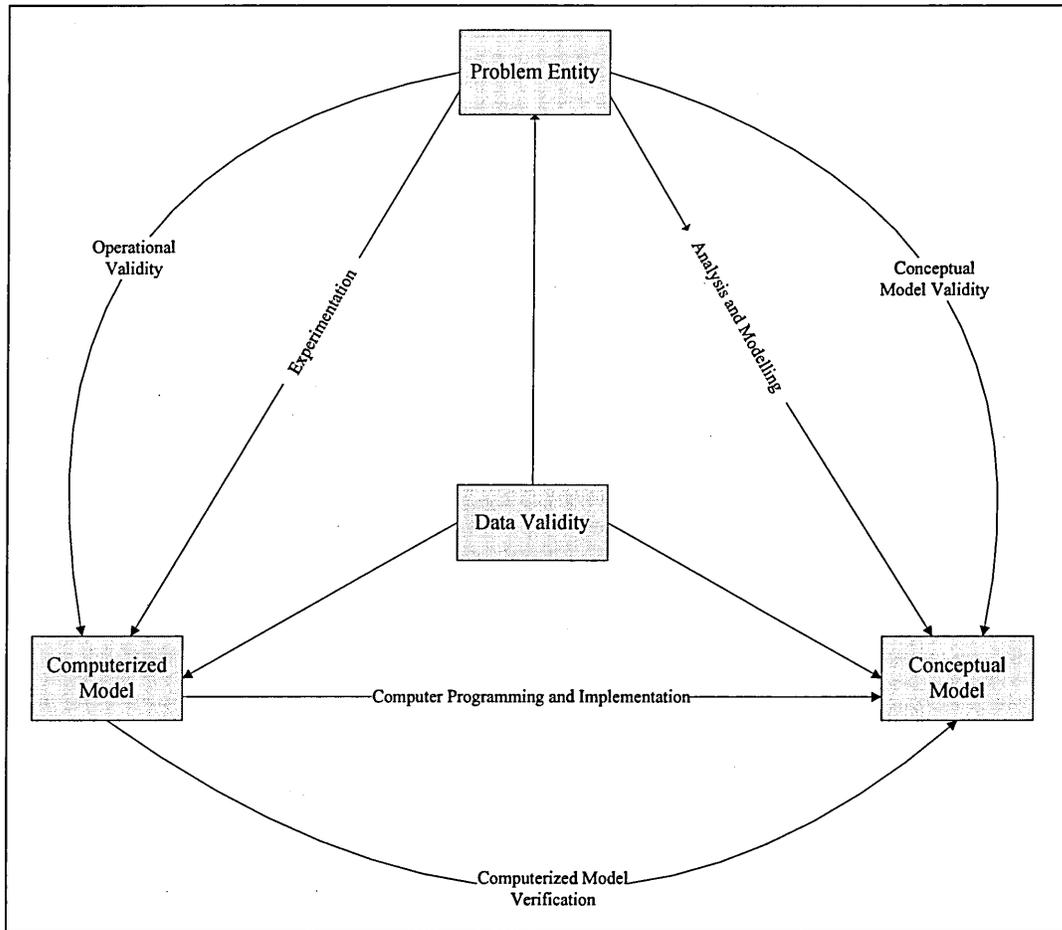


Figure 2.2: Simplified Version of the Modelling Process (Sargent,2000) .

According to Shannon (1998) what is needed is to design a model of the real system that neither oversimplifies the system to the point where the model becomes trivial (or worse misleading) nor carries so much detail that it becomes clumsy and prohibitively expensive to build and run. Simulation is both an art as well as a science. The programming and statistical components are the science part and the analysis and modelling components are the art (Shannon, 1998; Banks, 2000). One of the fatal errors made by a simulationist is building a model which will not represent the behaviour of the real system.

The model building phase of a simulation project has also been sub-divided in to various stages by different authors (See Table 2.2). Further Sargent (2000) has explained the model building process by using the diagram as shown in Figure 2.2.

Even though different authors have divided the model building process into between 2-5 stages, basically three generic stages can be identified, namely:

- **Conceptual model building;**
- **Computer model building;**
- **Model verification and validation.**

From Table 2.2 and Figure 2.2, it is apparent that a conceptual model is built before the computer model is built. A conceptual model is essentially a model where mathematical and logical relationships are defined (Oakshott,1997; Sargent,2000; Banks,2000). Conceptual model is seen as a model that is formulated completely independent of any programming language or simulation language (Arons, 1999). According to Mehta (2000) it is extremely useful to map out or structure the model on paper before building the model. The structure or map of the model can help to get an overview of the whole model and map out any of the complex areas where further investigation is needed. Practitioners are having the view that many of the pitfalls in the latter stages of the simulation model development process can be avoided by structuring conceptual model before building it on the computer (Robinson, 1994; Shannon, 2000; Mehta, 2000).

Once the conceptual model is built, it is translated into the computer model. A computerized model is the conceptual model implemented on a computer (Sargent, 2000). The computer model is actually a computer program. As noted by Trybula (1994), if the

logic is complex, the modeller may need to develop the model, run the simulation while tracing the code, and make modifications to the operation of the model.

After building the computer model, it is to be tested to find syntax and logical errors. According to Shannon (1998), this process is called verification and it gives the answer to the question “does the model work correctly?”. But validation is the determination that the model is an accurate representation of the real system. It provides the answer to the question “does the model behave the way the real world system does or will?”. Debugging a simulation model that is developed using a simulation language can be tedious and time consuming (Oakshott, 1997; Sadowski and Grabau, 1999; Sargent, 2000). Further it is an iterative process (Robinson, 1994) (See Figure 2.3). Several versions of a model are usually developed prior to obtaining a satisfactorily valid model (Sargent, 2000; Oakshott, 1997).

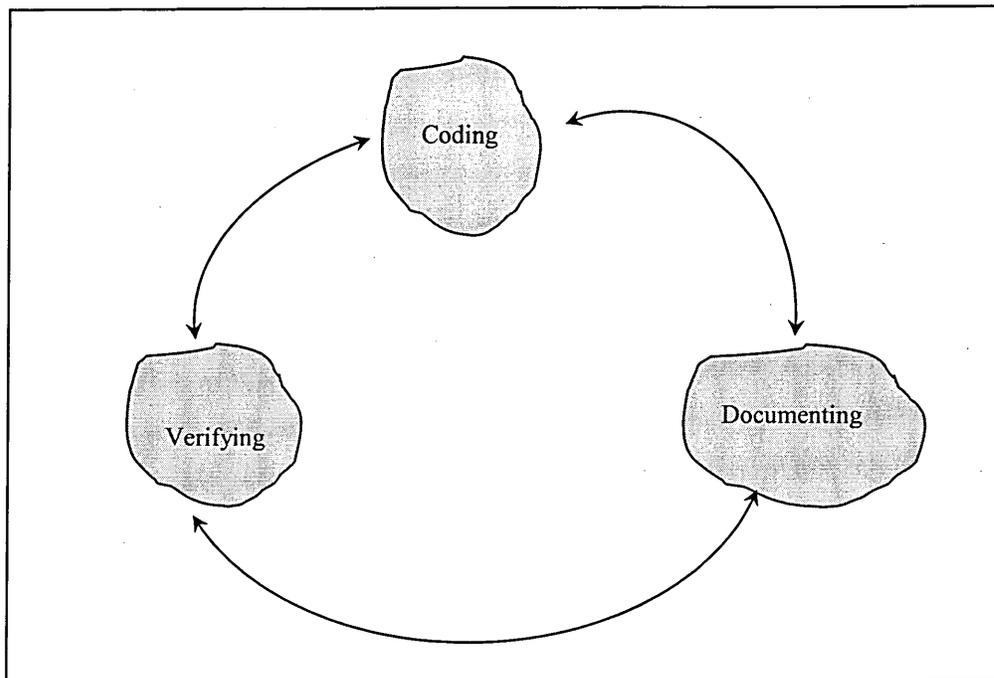


Figure 2.3: The model building cycle (Robinson, 1994).

Model building and experimentation are the most time consuming stages of a simulation project (Robinson, 1994) (see Figure 2.4). According to Sakthivel and Agrawal (1992), deriving a simulation model from an understanding of the system to be simulated is perhaps the most complex and time-consuming task of the simulation life-cycle. Other available literature also support the above view.(e.g. Trybula,1994; Umeda and Jones, 2001; Brown and Powers, 2000; Arons, 1999) (See Tables 2.3 and 2.4).

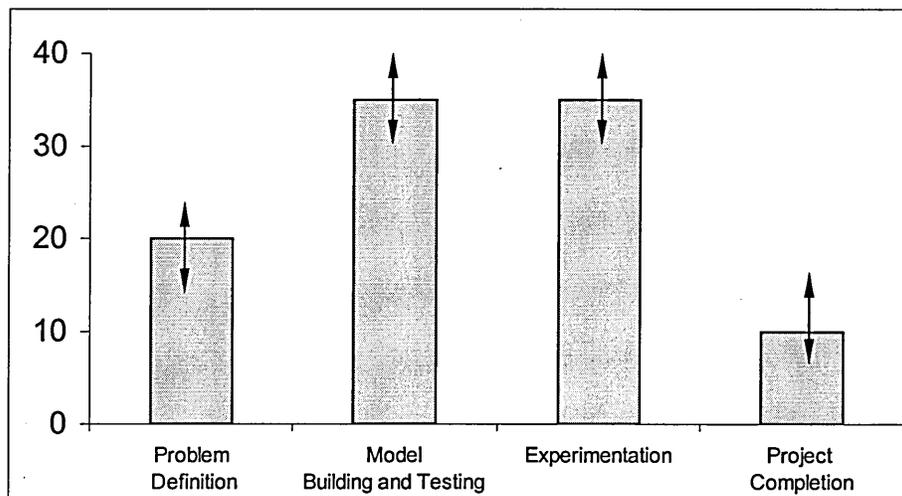


Figure 2.4: Project time –scales : a percentage breakdown(excluding implementation) (Robinson, 1994).

Therefore it can be argued that model building is a very time consuming phase of the simulation project development process. However, building a valid simulation model which represents the actual system is critical for the successfulness of a simulation project. A large number of tools are available to assist the modeller in developing a simulation model. Any mechanism that can assist in the development of a simulation model is called a simulation tool (Oakshott, 1997). Therefore a simulation tool is basically a computer

software which allows the modeller to develop the model and to make experiments. These software are explained in detail in the next section.

Stage	Percentage time
Problem Definition	~10%
Problem Analysis	~10%
Data Gathering and Validation	10% - 40%
Model Development	10% - 40%
Model Verification and Validation	~10%
Model Experiments	10% - 20%
Analysis of Results	~10%
Conclusion and Recommendations	~5%

Table 2.3: Time requirements for each stage of a simulation model based on personnel experiences (Trybula,1994).

Phase	Work_ load Ratio (%) [Range]
Data Collection/Gathering	20 [15 - 25]
Model Design	40 [30 - 60]
Animation	10 [5 – 15]
Model Modification	10 [5 – 15]
Simulation Experiments	10 [5 - 20]
Summary of Results	10 [5 – 10]

Table 2.4: Time requirements of each stage of a simulation project. Based on a survey conducted among simulation users in Japan (Umeda and Jones, 2001).

2.3 Simulation Software

As mentioned earlier, computer simulation refers to “methods for studying a wide variety of models of real world systems by numerical evaluation using software designed to imitate the system’s operation or characteristics, often over time” (Kelton et al., 1998). A simulation model is therefore a computer program which represents the logic of the system as entities with attributes arrive, join queues to await the assignment of required resources, are processed by the resources, released and exit the system. According to Shannon (1998), modellers have three generic choices in formulating the computer simulation model, namely:

- Build the model in a general-purpose language
- Build the model in a general purpose simulation language
- Use a special purpose simulation package (simulator)

Presently more than fifty different simulation software are available in the market from various vendors at different prices ranging from starting less than US\$50 up to US\$48,000 and almost all of them are PC based and Windows compatible (Swain,2001). Independent economic studies have estimated the size of the manufacturing simulation and visualization software market in the range of US\$650 million (NIST,2000).

Simulation languages allow the user to develop a model by writing a program using the constructs of the language. Simulators are packages that model a specific class of application. They are generally menu driven, requiring a user to input data and basic logic commands while little or no programming skills are required. As pointed out by Robinson

(1994), users of simulation languages are experts, central organizations and regular users while users of simulators are non-experts, distributed users and occasional users (see Figure 2.5).

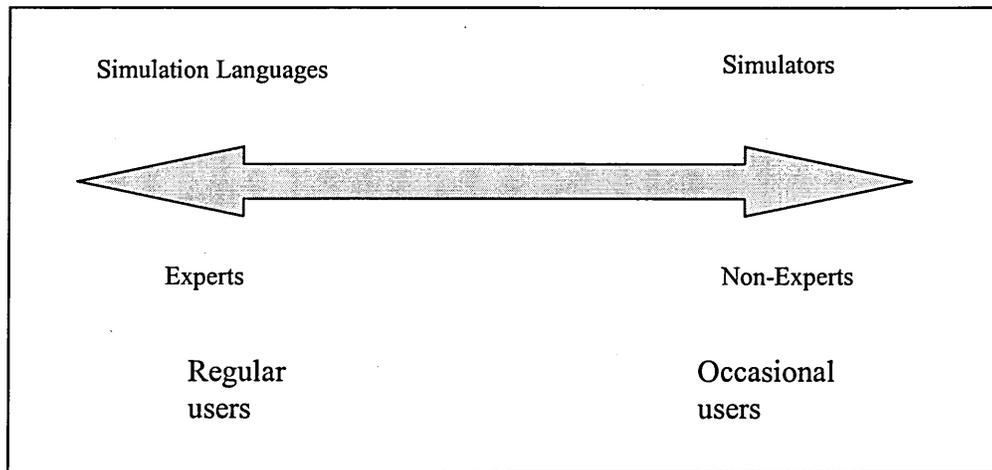


Figure 2.5: Simulation software preferences (Robinson 1994).

The main advantage of using a simulator is that the modeller does not need to spend time and effort on making models but the flexibility is not as great as the flexibility of a language. Programming-like commands and interfaces with programming languages are features which make a simulator flexible. But the distinction between simulators and simulation languages is blurring. They are moving towards each other by offering special features (Nikoukaran and Paul, 1999).

General-purpose languages such as FORTRAN, Pascal and C++ have been used by modellers in developing simulation models in addition to the simulation languages. Some of the common simulation languages are SIMAN, GPSS, SIMSCRIPT, SLAM and SIMFACTORY, WITNESS and EXTEND are some of the most popular simulators. Most

recent simulation languages and packages have been written in an object-oriented programming language (Oakshott, 1997).

With the wide spread use of Windows environment in PCs in 1990s graphical user interface (GUI) entered in to the simulation software. This involved the use of graphical objects or icons to represent parts of the model. The model could then be developed by linking these icons together in a logical fashion. They often incorporated a range of functions to assist in data input and output analysis (Oakshott, 1997). Modern simulation software such as ARENA and AUTOMOD combine the ease of use of a simulator with the power and flexibility of a simulation language (Kelton et al., 1998; Rohrer, 2000). A modeller can develop the model by using the ARENA graphical interface and ARENA generates the underlying SIMAN model (Takus and Profozich, 1997). However, according to Koh et al. (1996) simulation modelling still requires a high level of training, expertise and time, despite the availability of user-friendly simulators with their graphical use interface (GUI) modelling environment. For more detail and complex modelling, simulation languages are still preferred over simulators

Modern simulation software have features such as animation, real-time viewing, export animation (e.g. MPEG version that can run independent of simulation for presentation) (Swain, 2001). Animation is a powerful presentation and debugging tool that can greatly increase model clarity. Animation icons moving from block to block represent the flow of items through the system. Simulation software such as Arena allows the user to import graphics from other software such as AutoCAD and allows to link programme modules written in other languages such as Visual Basic and C++.

Even though simulators have received wide acceptance among simulation users a survey conducted among academic and industrial simulation users has shown that majority of them still use simulation languages as well (Hlupic, 2000)². (See Table 2.5).

Type of Software	Academics (%)	Industrial (%)
Simulators	83.3	55.5
Simulation Languages	61.1	22

Table 2.5: Percentage of different types of simulation software users among academic and industrial users (Hlupic, 2000).

Another reason for the use of general-purpose languages on simulation model development may be the high cost of simulation software. For example Arena standard version costs US\$ 13,500, AUTOMOD standard costs US\$15,000 and ProModel costs US\$ 18,500 (Swain,2001).

2.3.1 Features Expected from Simulation Software

According to Stanford and Graham (1998), until recently, discrete-event simulation tools were expensive, difficult to use, and limited in use to specialists within large corporations. However, in the last few years several low-cost (under US\$2000) discrete event simulation products have been introduced targeted for use by a broader range of simulation users than ever before. These tools take advantage of advances in computer hardware, software standards, graphical design and ease-of-use. Due to the diversity of simulation users,

² Simulationists' current practice of simulation software is shown in chapter six

different users expect different features from a piece of simulation software. Some of the most important features, as suggested by Oakshott (1997), are modelling flexibility (ability to use the software for any application), ease of use (ability to use a simulation tool with a minimum of learning time), animation (ability to watch the system being simulated in graphic detail), general simulation function (handling the generation of random varieties and specifying warm-up period etc.), statistical functions, interfacing with other software (such as spreadsheets and scheduling software), product help and support (on-line help, tips, good user guides etc.), price (inexpensive) and expandability. According to Mehta (2000), simulation software should provide tools that facilitate flexible modelling, ease of sharing the simulation efforts and effective utilization of the work already done in the past thereby avoiding the need of duplication of efforts. Research done among simulation users in Japan has shown that modelling support tools are considered as the most important feature when choosing a simulation software. User-friendly, icon-based modelling tools reduce the time and cost associated with building and maintaining simulation models (Umeda and Jones,2001).

The main three limitations of the simulation software as identified in user's surveys (Hlupic, 1999) are given in table 2.6.

Software limitations – 1992 survey	Software limitations – 1997 survey
Restricted flexibility	Limited standard
Slow	features/flexibility
Validation difficulties	Difficult to learn
	Expensive

Table 2.6: Major simulation software limitations (Hlupic, 1999)

In the same set of researches, when being asked about the success of modelling, 30.8% of participants in a 1992 survey had declared that they had been able to model desirable features of the system being modelled, 38.4 % had managed to model most of the features whilst 30.8% had problems. In a 1997 survey, 51.9% participants had declared that they had been able to model desirable features of the system being modelled whilst 48.1% had problems in modelling due to the software limitations and inflexibility (Hlupic, 1999).

In a panel discussion done at the 2000 Winter Simulation Conference on the topic *Simulation In The Future*, all 8 panellists (simulation consultants) emphasized the need for providing more modelling capabilities to users of simulation software. Their suggestions include tighter integration with other software, web based simulation, library driven simulation models as well (Banks,2000). So it is clear that there is a greater need for simulation software which provides efficient and simple methods for developing simulation models for simulation experts as well as non-simulation experts.

2.4 Summary

Discrete-event simulation is one of the most important tools available to engineers as well as to managers for analyzing the behaviour of large and complex systems. It is a multi-stage process and the most important stages of a simulation project are problem identification, model building, experimentation and implementation. Model building is the most time consuming phase among those phases. That phase consists of three stages: conceptual model building; computer model building; validation and verification. In building the computer model, a user can employ a general-purpose computer language, a simulation language or a simulator. Different users expect different features from simulation software,

but there is a greater need for simulation software which provides efficient and simple methods for developing simulation models for simulation experts as well as non-simulation experts. Research problems formulated to achieve the objectives specified in the first chapter and the research methodology are explained in the following chapter.

Chapter Three

Research Design

It was necessary to formulate a set of research questions in order to achieve the objectives specified in the first chapter. Finding the answers to those questions will ultimately enable the objectives to be fulfilled. It was very important to design the research appropriately to archive the objectives effectively. A number of methodologies for carrying out research and relevant concepts are discussed in the contemporary literature on research methods. However, the effectiveness of the research methodology employed depends on the context within which the research is carried out. Therefore, it was necessary to study the various concepts and methodologies before a suitable methodology was selected. Research questions formulated to achieve the objective stated in the first chapter are explained in the early part of this chapter. Research methodologies used in the present research are explained in detail in the latter part of the chapter.

3.1 Research Questions

The main objective of this research was to propose a new framework for rapid simulation model development. Three research questions were formed in order to achieve this objective. These three questions and rational behind the formulation of those questions are described below.

Question 1 : Why has the model development process become the most time consuming stage of a simulation project life cycle?

Literature on simulation revealed that model development was the most time consuming task among the stages of a simulation project life cycle¹. In order to accelerate this process it was necessary to know the reasons for lengthening the model development process. Research Question 1 was formulated for this purpose. Identification of those factors will enable the model development process to be accelerated by eliminating those factors or by reducing the impact of those factors on the model development process.

Question 2 : What are the alternative methods available for accelerating the simulation model development process ?

Once the above mentioned factors are identified it was necessary to understand the state of art relating to rapid simulation model development before proposing a new methodology. Therefore, the objective of this question was to find out different ways of accelerating the simulation model development process. To find the answer for this question, different research attempts of acceleration of simulation model development were reviewed.

Question 3 : Is there a need for a new methodology for accelerating the simulation model development process? If so, what is it?

It was necessary to critically review the existing approaches to find out whether there was a need for a completely a new approach. If that is the case, a new framework will be developed, otherwise improvements to the existing methodologies will be suggested.

¹ Results of the questionnaire survey also revealed the same (See chapter six).

Research methodology adopted in finding the answers for above mentioned questions is explained in detail in the rest of the chapter. However, a detail discussion on theories on research methodology is beyond the scope of this thesis. Since the present research is more descriptive in nature it was required to borrow some of the concepts from other fields such as sociology, medicine and psychology which are the areas where those kind of researches are frequently carried out. As a prelude to the selection of suitable research paradigm, some of the relevant concepts are explained briefly in sections 3.2 and 3.3.

3.2 Research Philosophy

According to Saunders et al. (2003) three views about the research process dominate the literature: positivism, interpretivism and realism. If positivism is adopted the researcher will prefer working with an observable social reality and that the end product of such law-like generalizations similar to those produced by the physical and natural scientists. There will be an emphasis on a highly structured methodology to facilitate replication and on quantifiable observations that lend themselves to statistical analysis. However, if the objective is to find the details of the situation to understand the reality or perhaps a reality working behind then the most suitable will be interpretivism. Realism, more related to social sciences, is based on the belief that a reality exists that is independent of human thoughts and beliefs.

3.3 Research Approach

According to Saunders et al. (2003) the main research approaches are *deductive approach*, a theory and hypothesis (hypotheses) are developed and a research strategy is designed to test the hypothesis/es, and *inductive approach*, data are collected and theory is developed as

a result of data analysis. The following table shows the differences between these two approaches.

Deduction emphasis	Induction emphasis
<ul style="list-style-type: none"> ▪ Scientific principles ▪ Moving from theory to data ▪ The need to explain causal relationships between variables ▪ The collection of quantitative data ▪ The application of controls to ensure validity of data ▪ The operationalisation of concepts to ensure clarity of definition ▪ A highly structured approach ▪ Researcher independence of what is being researched ▪ The necessity of select samples of sufficient size in order to generalise conclusions 	<ul style="list-style-type: none"> ▪ Gaining an understanding of the meanings humans attach to events ▪ A close understanding of the research context ▪ The collection of qualitative data ▪ A more flexible structure to permit changes to research emphasis as the research progresses ▪ A realisation that the researcher is part of the research process ▪ Less concern with the need to generalise

Table 3.1: Major differences between deductive and inductive approaches to research (Saunders et al., 2003)

According to Basili (1996), in the area of software engineering inductive paradigm might be best used when trying to understand the software process, product, people or environment. It attempts to extract from the world some form of model which tries to explain the underlying phenomenon being observed.

3.4 Selection of Research Paradigm

The success of a simulation project depends on various factors ranging from technical issues such as facilities provided by simulation software to organizational issues such as human interaction and top management commitment. Therefore in order to answer the research questions mentioned above it was necessary to explore the simulation model development process thoroughly and to understand the reasons leading to lengthening the process. Once those factors were identified next stage was to build a new framework, or theory, for accelerating simulation model development process. Therefore, the nature of this research was qualitative rather than quantitative. According to Patton (1990) qualitative methods permit the evaluator to study selected issues in depth and detail. Approaching fieldwork without being constrained by predetermined categories of analysis contributes to the depth, openness, and detail of qualitative inquiry. Quantitative methods on the other hand, require the use of standardized measures so that the varying perspectives and experiences of people can be fit into a limited number of predetermined response categories to which numbers are assigned. Qualitative methods typically produce a wealth of detailed information about a much smaller number of people and cases. This increases understanding of the cases and situations studied but reduces generalizability. According to Denzin and Lincoln (1998), Qualitative research is multi method in focus, involving an interpretive, naturalistic approach to its subject matter. This means that qualitative researchers study things in their natural settings, attempting to make sense of, or interpret, phenomena in terms meaning people bring to them. Qualitative research involve the studied use and collection of a variety of empirical materials-case study, personal experience, introspective, life story, interview, observational, historical, inetractional and visual texts-

that describe routine and problematic moments and meanings in individual's lives. In this context the most suitable **philosophy** for the present research was **interpretivism** and the appropriate research **approach** was **induction**. Once it was clear on the nature, philosophy and the approach of the research, next step was to identify the research strategy to be adopted in the selected paradigm.

3.5 Research Strategy

According to Saunders et al. (2003), some of the available research strategies are experiment, survey, case study, grounded theory, ethnography, action research, cross sectional and longitudinal studies and exploratory, descriptive and explanatory studies. Since the present research is qualitative in nature, and interpretivism and induction were adopted, it was necessary to select research strategies to reflect those characteristics in finding answers to the stated research questions. Therefore, the selected strategies were survey (e-mail questionnaire survey), case study and literature survey.

3.5.1 E-mail Questionnaire Survey

According to Saunders et al. (2003), questionnaires are usually not particularly good for exploratory or other research that requires large-number of open-ended questions. They can be used for descriptive or explanatory research. *Descriptive research* such as that undertaken using attitude and opinion questionnaire and questionnaires of organizational practices, will enable the researcher to identify and describe the variability in different phenomena. The objective of the first research question was to identify the reasons for lengthening the simulation model development process. Therefore, questionnaire was used

to obtain opinion about various statements built based on the literature and to know about the practices of different simulation practitioners and academics as a prelude for developing a new methodology for accelerating the simulation model development process. E-mail was selected against postal as the mode of questionnaire delivery due to three reasons.

Firstly, it is easy to reach a large audience throughout the globe with a e-mail survey. It is difficult to obtain postal addresses of simulation practitioners and academics, but there are several simulation user groups which maintain forums for simulation practitioners. In addition, e-mail directory of attendees of conferences such as Winter Simulation Conference is available on the web. Secondly, it is cost effective to conduct a e-mail survey because no postal cost is involved. Thirdly, 'Reply' facility in the mail software encourages the respondents to reply quickly. However, Sheehan and Hoy (1999) argued that use of 'reply' function of their e-mail programmes to return the completed surveys, write names and e-mail addresses on the electronic message the researcher would receive. While previous research has indicated that anonymity may have affected response rates positively, other researchers suggest that the lack of anonymity may not have any effect on response rates. However, in the present research anonymity is not a major problem as the questions asked are neither personal nor sensitive. Further, they have identified that a lack of national directory of e-mail addresses as a limitation of e-mail surveys. But as mentioned earlier, in the present research it was easier to obtain e-mail addresses than postal addresses of simulation practitioners. The following table shows the attributes of on-line surveys and postal questionnaire surveys.

Mode Attribute	On line	Postal
Populations characteristics for which suitable	Computer literate individuals who can be contacted by e-mail or internet	Literate individuals who can be contacted by post
Confidence that right person has responded	High if using e-mail	Low
Size of the sample	Large, can be geographically dispersed	
Likely response rate	Variable, 30% reasonable within organizations, internet 10% or lower	Variable, 30% reasonable
Feasible length of questionnaire	Conflicting advice; however, fewer screens probably better	6-8 A4 pages
Suitable type of questions	Closed questions but not too complex, complicated sequencing fine if uses IT. Must be of interest to respondent	Closed questions but not too complex, simple sequencing only. Must be of interest to respondent

Table 3.2: Attributes of on line and postal questionnaire surveys

The population characteristics given under the on-line survey exactly fit to the population of simulation practitioners and academics. Therefore e-mail survey was more suitable in the present research against postal questionnaire survey. However, since the length of the questionnaire was kept at the minimum to increase the response rate and there was no interaction with the respondents, other methods were also applied in conjunction with the questionnaire method. The questionnaire survey and results are explained in detail in Chapter six.

3.5.2 Case Study Research

Case study technique is used in two stages of the research process. Firstly, as a tool of finding answers to ‘why’, ‘what’ and ‘how’ questions, or understanding the situation and secondly as a tool of validation of the proposed technique. As noted by Zelkowitz and Wallace (1998), in a case study, researchers monitor a project and collect data over time. Yin (2003) provides a two part definition for a case study.

1. A case study is an empirical inquiry that

- investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident.

2. The case study inquiry that

- copes with the technicality distinctive situation in which there will be many more variables of interest than data points, and as one result
- relies on multiple sources of evidence, with data needing to converge in a triangulating fashion, as another result
- benefits from the prior development theoretical propositions to guide data collection and analysis.

Further he identifies three common concerns about the case studies. Firstly, lack of rigor of case study research. Secondly, case studies provide little basis for scientific generalization. Thirdly, they take too long and result in massive, unreadable documents. He strongly argues against these concerns and proves that case study is also a solid research strategy as other strategies such as experimentation.

3.5.3 Literature Survey

The last strategy used for finding answers for the stated research questions was literature survey. Reasons for lengthening the simulation model development process and alternative approaches to accelerating the simulation model development process were found from journal articles, conference proceedings, electronic database, theses and books on simulation. Since those materials are authored by experts in the field, they are reliable and may provide a direction for a new methodology for accelerating the simulation model development process.

3.6 Development of New Rapid Simulation Model Development Framework

Once the data were collected by applying the above mentioned strategies, they were critically reviewed to find the answers for research questions. If it was found that the present approaches were not the best for accelerating the simulation model development process a new methodology would be proposed based on the facts collected. Once the new methodology was proposed it was validated using three strategies. The new methodology and the validation process is discussed in detail in Chapter eight. Figure 3.1 provides a comprehensive view of the research design.

3.7 Summary

In the context and the nature of the present research it was found that the most suitable research philosophy was interpretivism and the most suitable research approach was induction. In this context the strategy used for research in order to find the answers to the formulated three research questions were case study, questionnaire survey and literature survey. A detail description of three case studies done in order to find the answers to the first research question are given in the next chapter.

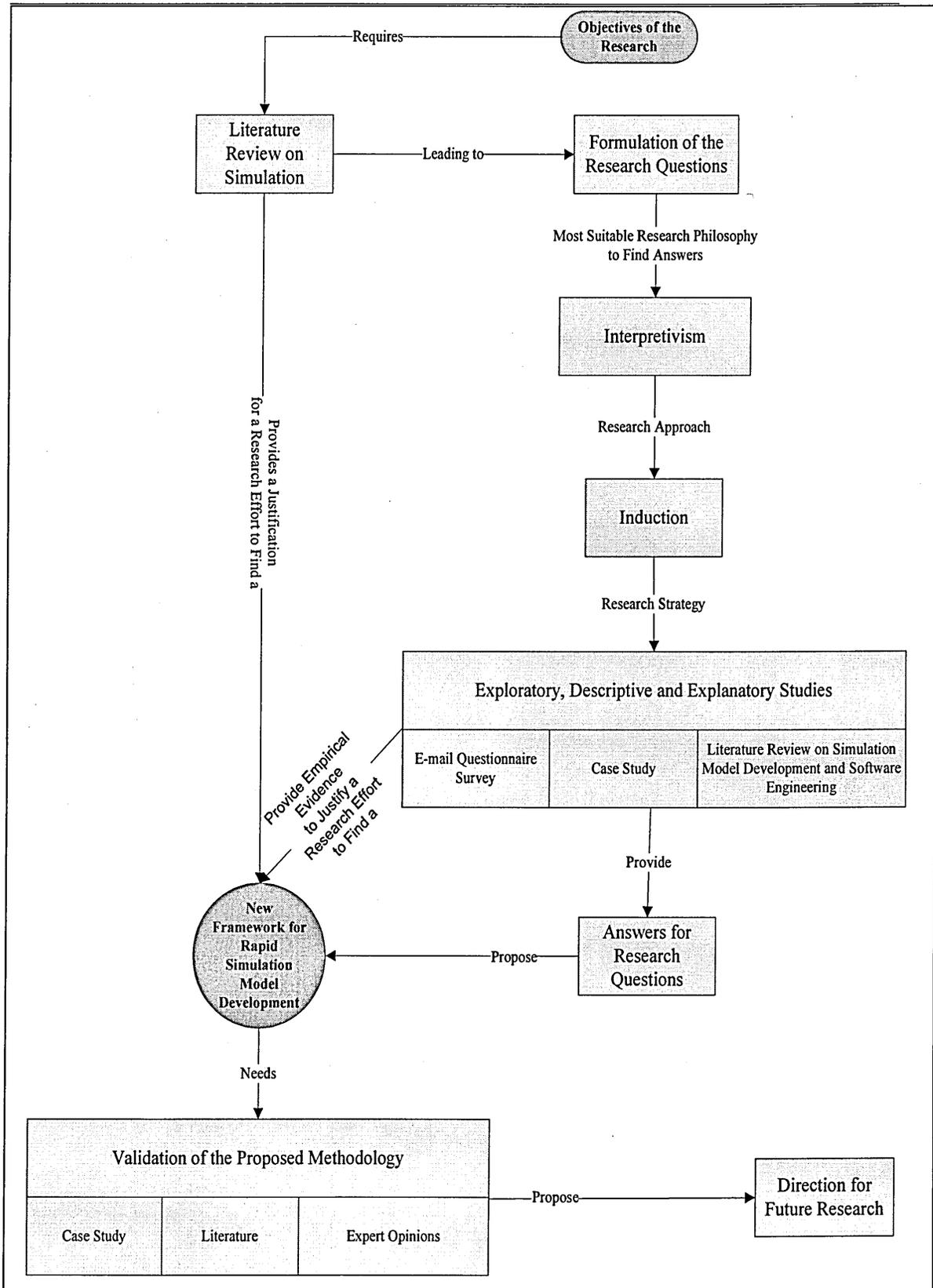


Figure 3.1 : A Comprehensive View of the Research Design

Chapter Four

Case Study Research

One of the research strategies which was identified in the third chapter was case study research. The same set of case studies were used for three different purposes in this research. Firstly, to find out reasons for lengthening the simulation model development process. Secondly, to obtain an idea about the direction of the new methodology for accelerating the simulation model development process. Thirdly, once the new methodology is proposed, it is applied to these cases in order to validate the proposed method. This chapter explains three case studies carried out¹ and presents the facts gathered through discussions held with the consultants involved in each of the selected cases. Lessons learned from the case studies when they were taken as a whole are explained at the end of the chapter.

4.1 Background of the study

Three recently concluded simulation projects which were undertaken in manufacturing organizations were selected as cases for study. Detailed structured discussions were held with the consultant for each case study with the objectives mentioned above in mind. The typical situation of each case was that a number of different parts move through a system of machines. In each case, only one consultant was involved in the model development process from the beginning to the end.

¹ Some real time data are not presented to preserve client confidentiality

4.2 Case Study A

In this study the client wanted to develop two simulation models for two scenarios of a printing press. This press has not yet been installed and the client wanted to develop two simulation models, one for a push system and one for a pull system. There were 7 machines, one inspection, one common assembly station and automated storage and retrieval system in the facility (See Figure 4.1). This company produced 10 products by assembling 35 different components according to bill of materials (BOM). Each component follows a different route through the facility and at the each machine there is a set-up time and a process time for each component. Parts and the final products are moved through the facility by using two automatic guided vehicles (AGV). Since, the client's office located 150 miles away from the consultant's office, communication was mainly done through telephone, e-mail and fax.

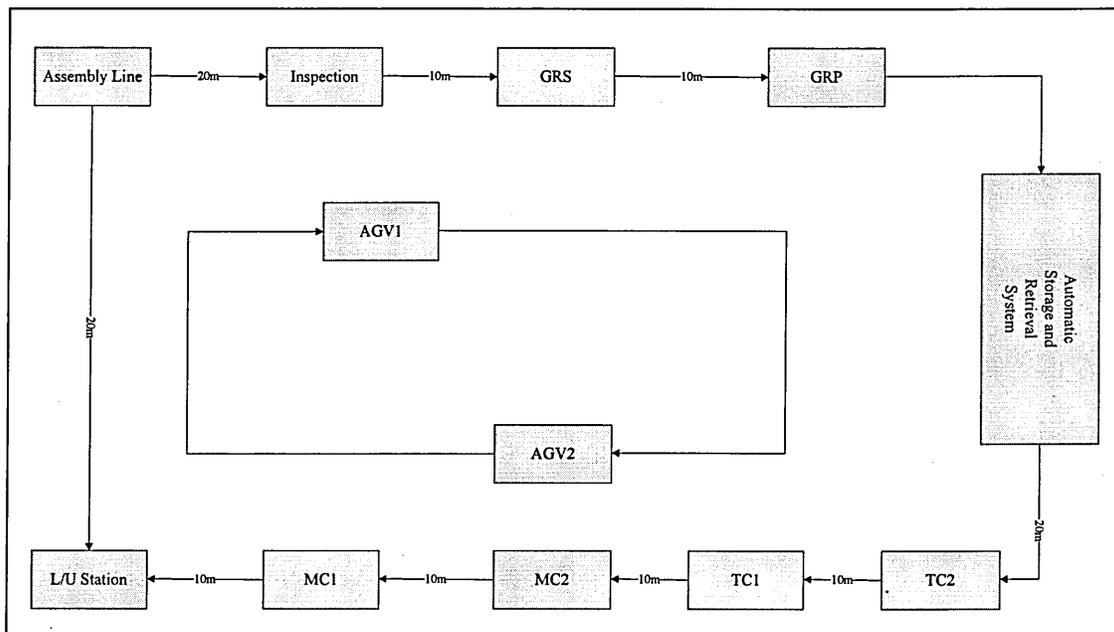


Figure 4.1: Layout of the printing press.

Client provided the layout plan of the proposed factory which is shown in Figure 4.1, BOMs in the form of a tree diagram, Route for each part in a table, setup and process time and assembly times for each part on each machine in the same table and a description about the processes to be simulated. (See Figure 4.2)

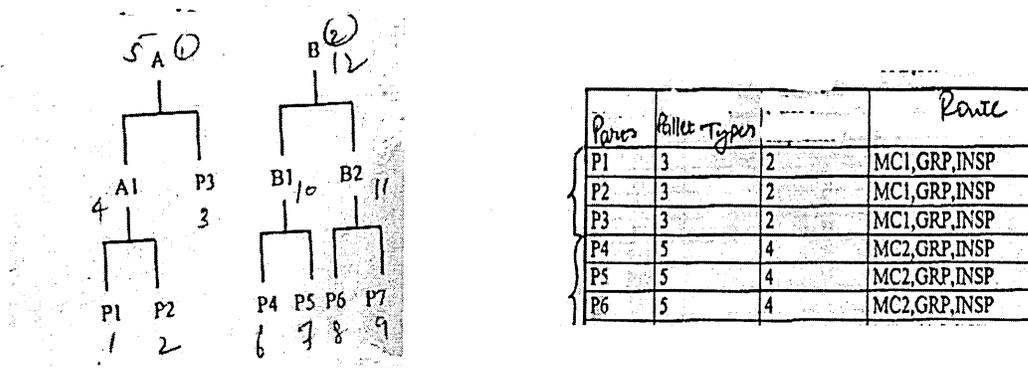


Figure 4.2: Some of the information provided by the client

4.3 Case Study B

The client in this case was a manufacturing company producing compressor casings. This company was recently acquired by a large company based in Germany. Due to the acquisition, the manufacture of the compressor casing is now considered one of the core competencies in this site. The equipment currently used to machine the compressor casing is more than 20 years old and considered outdated. This is significant as it is anticipated that production will increase four fold by 2008. The purpose of this project was to evaluate the consequences of purchasing new equipment for machining the casings. Since the problem was too complex to resolve manually, the client believed that a simulation model would give a true dynamic representation of how the Compressor Casing Cell performs based on all real life variables such as queue times, down times, repair times, manning availability and so on in addition to enabling what-if analysis. The objective was to determine the performance of the current system and compare it against that anticipated

once new equipment is purchased. Two simulation models were produced, the first representing the current facility the and second representing the future state.

Eight parts went through the system namely; RM13211, RM13011A, RT13211, RW13211A, MW13011, RM13012, RT13012 and RW13012A. These parts were subjected to operations Lathe, Bandsaw, Stress Relieve and Shotblast, Mazak, Fitting, Lathe, Dye Pen, STS and Final Inspection. There were situations where the same part went through the same machine more than once to perform two different tasks. The following table shows the route of the part RT13012 - HP Casing through the system.

Operation Number	Machine / Operation Description
B	Lathe
C	Bandsaw
D&E	Stress Relieve & Shotblast
F	Mazak
.	.
.	.
.	.
.	.
.	.
K	Fitters
L	Dye Pen
M	STS
N	Final Insp

Table 4.1: Route of the part RT13012 –HP Casing through the system.

A similar kind of table for each part was developed by the consultant after discussions held with the client. The following table shows how the responsibilities were divided between the client and the consultant and the agreed time table.

From – To	Activity	Responsibilities
Week 1	Identify the project scope and complete the project proposal.	Client : Sign-off model specification.
Week 2	Collect and analyse operating procedures and data.	Consultant : To identify the data required for the model Client : To provide the basic data sets to initiate model building including operating logic.
Week 2	Build simulation models	Consultant : Model development. Client : to provide sample historical order data to assist in building and validating the model.
Week 3	Conclude build and Validate the models	Close co-operation between client and consultant to ensure validity of model.
Week 4	Present Results	

Table 4.2: Time table of the project agreed between the client and the consultant.

4.4 Case Study C

This model was developed for the same company described in study B. But the consultant had to work with a different set of people from another section of the company. The objective of this project was also to assess the cost saving arisen due to planned purchase of

machines. The consultant developed two simulation models for the present layout and for a future proposed layout.

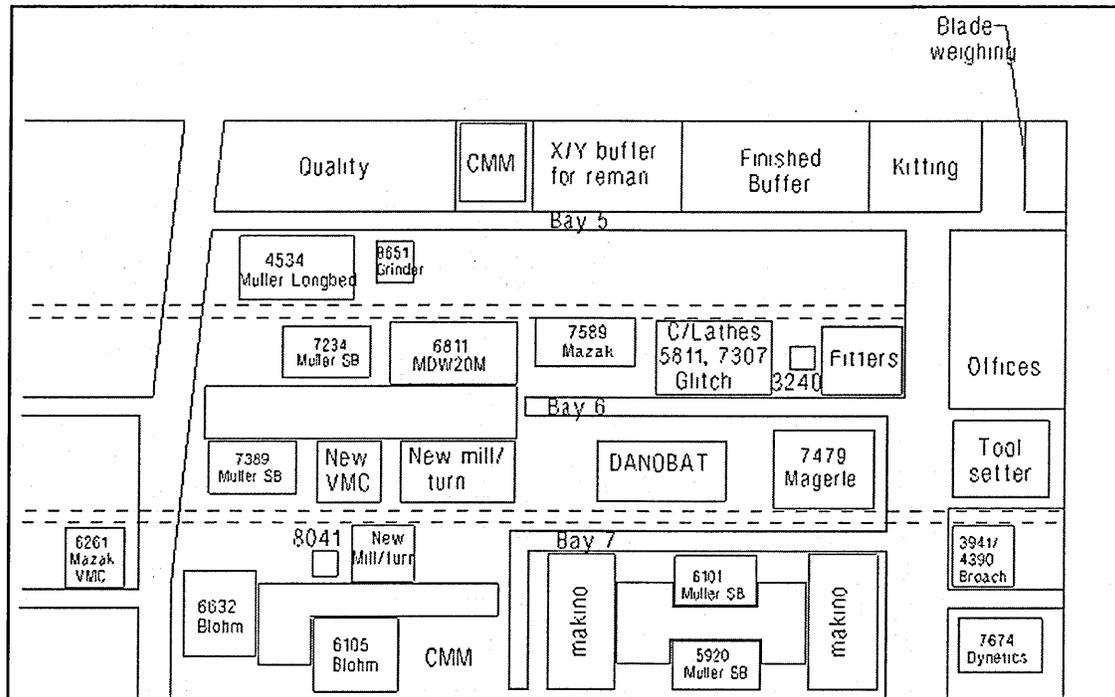


Figure 4.2: Layout of the modelled factory

In this system thirty-two parts move through the system namely. These parts had very similar but different names such as MW11261A, RW11261A, RM11261A and RT11061B. Current route of Part MW11261A is shown in table 4.3 as an example. The client provided similar set of data for all the parts. Further the client provided layout of the factory which shows in Figure 4.2.

OP	M/C	(MINS) SETUP	(MINS) M/CING	OPERATION DESC.	(DAYS) LEAD
70	MDW20M / LB	60	120	RGH STEM END	0
80	MDW20M / LB	60	120	RGH DISC END	0
90	STRESS RELIEVE	0	0		1
100	MDW20M / LB	85	400	S/FIN DISC END	0.75
110	MDW20M / LB	90	375	S/FIN STEM END	0
120	MAZAK	130	105	DRILL & TAP HOLES	3
130	FITTERS	5	20	DEBURR & FIT INSERTS	0.5
.
.
.
.
.
.
.
.
.
.
.
.
280	CMM	0	0	HIRTH TEETH	0.75
290	FITTERS	0	45	DEBURR HIRTH TEETH	0.5
300	MAG PART	0	0		1
310	INSPECT	0	0		0.5

Table 4.3: Current route of part MW11261A.

4.5 Summary of the Discussions

At this phase of the case study, the main objective was to find the reasons for lengthening the simulation model development process. In order to achieve this objective detail structured interviews were held with the consultants involved in the projects selected for case study. Following are the answers provided by them.

Question 1: How was the project carried out from initial discussion to model delivery ?***Study A :***

After the initial discussion the client provided information such as plant layout, BOMs and part routes. Then the consultant verbally explained to the client what was going to be done. The consultant keeps informing the client of the progress of the project to during the model development process. Once the model is finalized, it is delivered to the client with a written description of the logic of the model. Then the client proposed some changes and they were done.

Study B :

After an initial discussion the consultant provided a project proposal. Once it was approved data collection was started. Then the consultant developed the computer simulation model in his office and provided it to the client for validation. A meeting was held with a team of people those who were going to evaluate the model. A detail schedule for this case is given in Table 4.2

Study C :

The initial discussion held about six months before the actual project started. Then after few months the client wanted to start the project. Then the consultant had few discussions with the client and provided the specifications. However, a formal schedule was not agreed between the consultant and the client as in Case B. Then the consultant developed the model and submitted it to the client. However, it was necessary to make several changes to the developed computer model until the client satisfied.

<i>Question 2: Could you keep up the agreed schedule?</i>
<i>Study A :</i>
Not really. But somehow it was possible to deliver the project within a delayed but mutually agreed time period. However, it was necessary to introduce a few assumptions to make the model simple and thereby to reduce the development time.
<i>Study B :</i>
Yes. The client was very much focused on the project and they were desperate to complete the project on time. Therefore they were ready to accept the model developed by the modeller without much hesitance.
<i>Study C :</i>
Initial schedule was to complete the project by Christmas. As project progressed the date was brought forward to early November which it could not be met. A new date of mid-November was agreed. But it was not possible to meet due to the fact that the client was on holiday. So project completed early December.

<i>Question 3: Did you keep contact with the same person/s through out the project?</i>
<i>Study A :</i>
Yes
<i>Study B :</i>
Yes

Study C :

Yes. But towards the latter part of the project it was necessary to contact the manger of the normal contact person because he was on holiday. In the discussion the manager clarified the number of machines.

Question 4: Do you think that the client (persons you had contacted) had a good understanding of the system to be modelled? What they expect from the model?

Study A :

Not really. What the client knew was that she needed to compare Push and Pull models for a factory installation. In that sense the client was clear about the objective of the project, that a comparison between push and pull model should be done. But the client did not have a good idea about the processes which are to be included into the system.

Study B :

Yes. The client was very focused on the capital cause. The client had indicated that they would purchase a new machine from a supplier but had not yet signed a formal agreement. The client was desperate to get a cost comparison for the present system and to be system before signing the formal agreement with the supplier of the new machine.

Study C :

No. They really needed to conduct a simulation study for the factory before purchasing the new machines. There was enthusiasm within the organization regarding using

simulation as a tool for comparing costs of different strategies. They even had a plan to recruit a full time simulation consultant to the company. However, the person who had a contact with the consultant did not have a detailed knowledge of day to day activities of the shop floor. The consultant did not meet the shop floor workers and discuss. Neither the client nor the consultant thought that there was such a need.

Question 5: How did you know that you had developed the correct model? Or in other words how did you validate the developed computer simulation model?

Study A :

The consultant submitted a long description of the logic behind the developed computer model along with the computer simulation model. The model was analyzed by a person who is having a knowledge about the simulation software. Validation was done not by the person who had regular contact during the model development process.

Study B :

Model verification was done by comparing the mathematically calculated values against the values created by the computer simulation model. But there was no way of ensuring the validity, i.e. whether the developed model represent the actual model because neither consultant nor client knows the exact details of the simulated system. However, the objective of the study was to compare the present system with the future system. Therefore, the consultant believed that the developed model was adequate to serve that purpose.

Study C :

Same as in case B.

Question 6: Was it necessary to make changes to the developed computer simulation model?

Study A :

Yes. It was necessary to make changes because the developed computer simulation model was finally evaluated not by the normal contact person. He did not agree with the consultant's view on push and pull models and it was necessary to make major changes to the computer simulation model.

Study B :

Yes. As mentioned in Question 1, it was necessary to make changes to the built computer model. Even though the consultant kept contact mainly with a one person from the client organization validation of the built model was done by a team. The consultant explained the logic of the computer simulation model and his understanding of the system to this team in the middle of the project by using an incomplete computer simulation model.

Study C :

Yes. After developing the model it was found that some of the machines which were included into the model were not really needed and vice versa. Another problem was some of the machines which were shown on the layout did not really exist at the time of model delivery. At this stage it was found that some processes are having interactions with other parts of the factory. However, these interactions were not been mentioned in the previous discussions. Same as in study B, validation was done by a team.

Question 7: What do you think were the reasons for lengthening the model development?

Study A :

- i. Difficulty of working with different but very similar routes for 35 parts.
- ii. Lack of vision and knowledge of the client. Client wanted to compare push and pull models for the same factory. But she did not precisely know the differences between the push and pull models.
- iii. Very time consuming modelling process. For example there were 28 decisions, 35 creates etc in the final computer simulation model.
- iv. Inclusion of processing times for 35 parts into the computer model took a lot of time. Each part had a process time and set-up time at each machine.
- v. Changes proposed at the validation stage

Study B :

- i. It was necessary to change the built computer simulation model. The consultant did not have an opportunity to see the proposed machine hence did not appreciate that it had three machine beds to enable set up of another part while machining the current part.

Study C :

- i. Confusing parts and machines names made the job difficult. Parts have very similar names such as MW11261A, EW11261A and RM11261A. Even though one name was given for a certain machine in the information provided by the client, it was found that different machine names are used in practice.
- ii. Misunderstandings about the system to be modelled. For example some of the

machines which the consultant thought that were inside the cell found to be outside the cell later. In a later stage it was found that the provided setup times were for batches of parts whereas the machine times were for individual parts.

- iii. Layout and part routes which were provided early stages of the model building process were changed later. Scope of the project changed continuously throughout the project.
- iv. A plan of the layout of the factory was initially provided. However, later it was found that the provided plan was not up-to-date. Layout provided gave details of machines both inside and out of cell. So, although 5 short bed machines on the plan were drawn, only 2 were used by the cell.
- v. Amalgamation of quantitative model was difficult due to some problems found with the simulation software. For example creation of Duplicate parts produces misleading results such as work-in-progress and number of parts in the system.

Lessons learned by analyzing all three cases together are presented in the following section as a prelude to answer the research questions in section 4.7.

4.6 Lessons Learned

Due to the nature of the case study research it is not possible to draw definite conclusions. However, the objective of the case study was to make a detail empirical analysis on the

simulation model development process and thereby to understand the reasons affecting the lengthening of the process.

In all three studies, the modeller kept contact mainly with one person with in the client organization through out the project. However, due to some reason when he had to meet another person, their (the new contacts) view on the system was different from that of the regular contact's view. Even though the required data are provided by one person it seems that the validation/evaluation of the computer model is done by a team. The modeller may develop the compute model based on the data provided by one person. But once it comes to the validation, the persons who are involved in the validation may propose changes to the developed model because they may have a different perception about the whole system or part of the system.

It appears that in all cases the client has a need for developing a computer simulation model for a specific task, for example compare push and pull model or compare the cost of a new machine set up versus the existing set up. Further, the clients had the commitment for the project. However, when it comes to finding out what is to be modelled it seems that the client and the modeller are not in the same platform. It appears that once the available data are provided, the client thinks that it is the consultant's responsibility to somehow come up with the computer simulation model of the system. But again when it comes to the validation of the model, then the persons from the client organization raise various questions about the developed model. It is an indication that the client is in a better position to explain his requirements when some model is in hand.

In all three cases it was necessary to make changes to the developed computer model even though the problems arisen during the model development had already been solved with the client. It was not easy to come to a common understanding of the system to be modelled. This happened not only because the lack of understanding about the system of the modeller, quite obvious due to the consultant's lack of exposure to the system, but also because of the clients lack of understanding of the system or client's inability to express his understanding the modeller. This led to the development of models which will not represent the actual system.

In all the cases, most of the times information which clients have provided were from pre-existing data, i.e. the data which had been produced for some other purpose. This data may not represent the situation of the system at the point of modelling. For example a blue print of the factory drawn some time ago may not reflect any changes in the location or type of equipment available. Even though these documents are a great help to the modeller, these kind of secondary data will not guarantee the production of the correct model.

A typical characteristic of these cases was that there was a time between the initial discussion of the project between the client and the actual start of the project. During this period scope of the project may have been changed. However, when the deadline approaches, the client pushes the consultant too hard to finish the project. But still again when it comes to the validation stage, client wants to make various changes. Therefore, from the information gathered from the cases it seems that in practice a typical simulation project is carried out in the following manner.

-
- Initiate the project, and then a period of silence followed by the client suddenly wanting the project to be completed as soon as possible.
 - The modeller visits the client's office several times and gets an idea about the system to be simulated.
 - The client and the modeller agree in writing or verbally about the system to be simulated.
 - Then the modeller develops the computer simulation model at his office while keeping contact with the client to get problems solved which arose during the model development process.
 - Modeller submits the computer model to a team of people from the client organization for evaluation.
 - Modify the developed computer model to reflect the proposed changes during the evaluation process

It appears that when at the early stages of the project, i.e. when the deadline of the project is reasonably away, clients want to make more changes and demand various alterations to the scope of the project. However, when deadline approaches clients are willing to accept a less-than-perfect model or to accept more assumptions about the model. Client and modellers lack of understanding of the system makes the validation task difficult.

4.7 Summary

Three case studies carried out as a part of the research were explained in detail in this chapter. These case studies are used to find answers to the research questions, and to find a

direction for a new approach for accelerating the simulation model development process and also to validate the proposed methodology. Findings of the research are explained in the next chapter.

Chapter Five

Findings

The previous chapter explained case studies carried out in detail. This chapter presents the findings of the research, particularly answers to research questions 1, 2 and 3 formulated in Chapter Three. Firstly, the answer to the first research question is found from literature and case studies. Secondly the current status of research on rapid simulation model development is discussed in order to identify the alternative approaches to accelerate the simulation model development process. Finally, the available approaches are reviewed to identify the weaknesses and the strengths of them to find out whether there is a need for a new methodology for accelerating the simulation model development process.

The first research question which was formulated in the third chapter is “**why has the model development process become the most time consuming stage of a simulation project life cycle?**”. In order to answer this question it is required to find the reasons for lengthening the simulation model development process. Firstly, these reasons are found from the available literature.

5.1 Reasons for Lengthening Simulation Model Development Process

According to Benjamin et al. (2000), only a small fraction of the potential practical benefits of simulation modelling and analysis have reached the potentially large user community because of the relatively high requirement of time, effort and cost needed to build and successfully use simulation models. Further, simulation suffers a lack of wide acceptance by decision makers due to a number of factors including a) the semantic gap between the

description of a system internalized by the decision maker and abstract model constructed by the simulation modeller, b) the relatively long lead times and communication efforts required to produce a simulation model and c) the extensive training and skill required for the effective design and use of simulation modelling techniques.

As pointed out by So and Lew (1999), to ensure the correctness and validity of the simulation model, thorough knowledge of system domain, simulation methodology and programming language [simulation tool] needs to be consolidated. The modeller may have extensive domain knowledge and a clear view of the system to be modelled. However he may have little experience with simulation tools (Arons and Asperen,2000). The introduction of low-cost simulation tools has diversified the customer base for discrete-event simulation (Stanford and Graham, 1998). According to Umeda and Jones (2001) recent advances in number of technologies have provided industrial users with high performance computer hardware and graphical user interfaces (GUI). These advances have made possible to run simulation tools on desktop computers. Rathburn and Weinroth (1999) have identified two categories of impediments to providing *desktop simulation* to managers. First, the manager's lack of technical skill in the development of the simulation code often limit the ability of the decision maker to develop appropriate models. Second, current simulation model development methodology discourages exploration of variations in the model. Specially with the entry of non-experts into the simulation field in areas such as business process re-engineering, a gap had been created between domain experts, those who are having a good knowledge about the system to be modelled, and simulation experts, those who are modelling and analyzing the system. According to Benjamin et al. (2000), recent advances in the area of simulation modelling represent important advances for

improving the productivity of simulation modellers, but do little to aid the non-simulation trained-decision maker.

It is explained in chapter two that there are evidences to show that model building is an iterative process. According to Johnson (1999) often when simulation is used to validate a system design, the design itself will change multiple times. This in turn forces the model to be reworked, or sometimes even recreated. Sometimes the modeller has to even re-work the model from scratch.

According to Arons (1999) modelling is one of the most difficult and time consuming parts of the simulation process. As per Son et al. (2000) building, running and analyzing a simulation model can be a time-consuming and error-prone process. According to Pitts and Hwang (1999) simulation languages such as GPSS/H require the user to understand not only statistical data but also the language and its unique program flow. Even the graphical capability of the AWESIM language is cumbersome and often difficult to difficult to understand for even an individual familiar with programming. According to So and Lew (1999) formulation phase of an abstract model and the computer programming for simulation often involve lengthy and costly procedures.

Answer found for the research question from the available literature was explained above.

Answer found for the same question from the case studies is explained below.

From the discussions held with consultants involved in the cases explained in the previous chapter it was apparent that there was a gap in understanding between the model envisaged by the user and the model perceived by the modeller. This led to rework and hence

lengthening the model development process. This happens not only because the consultant's lack of understanding of the system but also client's lack of understanding of the system or client's inability to express his understanding to the client. However, when it comes to the validation, i.e. when there is a model to discuss, the client is in a better position to comment about the system. The process is further delayed by the fact that the validation is done by a different set of people. Involvement of a several persons in the validation process helps to develop a better quality model representing the system. But as mentioned earlier, difference of perceptions regarding the system held by different people, especially surfaced after developing the computer model, may lead to lengthening of the model development process.

Programming difficulties were also a major factor leading to delays in simulation projects. Even with the advancements in the simulation software, modelling is still a very time consuming task. Not only modelling the logic of the system, but amalgamating the quantitative data into the model is also time consuming.

Therefore from both the analysis of literature and case studies it is clear that **factors leading to the lengthening of the simulation model development are the lack of understanding of the system to be simulated between the user and the modeller and the difficulty of programming.**

Once those factors are identified, the next step is to find out how different researchers have tried to accelerate the simulation model development process. This will provide the answer

for the second research question i.e. **What are the alternative methods available for accelerating the simulation model development process ?**

5.2 Rapid Simulation Model Development: Current Status

Analysis of research into rapid simulation model development shows that most of the researchers still try to find answers to the following questions raised by Centeno and Standridge in 1992.

- Could simulation modelling be done without the analysts being concerned with the programming language?
- Is it possible to reuse simulation models?
- Is it possible to connect existing information systems to the simulation modelling system?
- Is it possible to simplify the model construction process?

In finding answers to the issues raised above, topics such as reuse of the simulation models, library driven simulation, i.e. the ability to use a library of pre-built icons during the model building process, and integration of simulation software with other software such as Excel and Access are in discussion among researchers. Those different approaches in accelerating simulation model development process are explained in the next section.

5.2.1 Model Reuse Approach

Among different approaches, model reuse approach is first discussed. Reuse generally means that previously designed and/or implemented elements are used again (and again) in

later projects with or without changes (Kovács et al.,1999). Those who advocate the re-use approach adopt the following propositions. The simulation models created are typically of the “analysis and throw away” type. In other words these models were developed, validated, experimented and filed away (Koh et al.,1996). Often multiple models are built to simulate similar systems that have small differences (Brown and Powers,2000). In the field of simulation of manufacturing systems this issue often occurs, when different systems with some similar features have to be managed. According to Kovács et al. (1999) the basic components of different FMS and FMC (Flexible Manufacturing Cell) are the same type of machine tools, robots, transfer equipment etc. In the relevant aspects they usually differ from each other only in their quantity and working parameters. As pointed out by Arons (1999) “the modeller constantly has the feeling that he is reinventing the wheel again and again despite the advance tools provided by simulation languages. Did he himself or someone else not build a similar model or sub-model in the past? Was such a model not build somewhere else? And if so how could he retrieve this model in a systematical manner?”. A Number of different methods of reusing models are found in the existing literature.

5.2.1.1 Model Reuse: Library Driven Approach

Mertins et al. (2000) presented an approach of using template libraries for developing simulation models. Under this approach a template can be seen as a building block of a huge simulation scenario, or as a pre defined component of a manufacturing design, a transport system, a warehouse etc. The user can build the model by parameterization of the building blocks. The advantage of the template library is that this approach includes a

mechanism to reuse simulation models from different simulation scenarios. But the disadvantage is that hard programming is needed to integrate different simulation models.

Benjamin et al. (2000) have discussed a method called MODELSIM. There are three levels of libraries in this method; Domain Analysis, Model Specification and Execution and Analysis, and three kinds of tools; Domain Description Tools, Simulation Design Tools and Execution and Analysis Tools in this architecture. Domain knowledge is stored by using IDEF process models and IDEF ontology models. Once an appropriate model is selected from the domain model library, firstly the conceptual model and secondly the detailed simulation model are developed by using IDEF3 process modelling language for conceptual simulation model development (I3CML). Then the simulation code in different target simulation languages is automatically generated. Other tools allow the user to design experiments, conduct statistical analysis and do simulation-based optimization. Execution and Analysis tools refer to the collection of component-based tools consists of Simulation Engine, Experiment Analyzer, Output Analyzer and Optimization Engine. The components of the architecture are being developed in Visual Basic and C++ using Microsoft's OLE, COM+ and ActiveX technologies. The researchers argue that this technology will significantly reduce time, effort and cost required to develop, deploy and maintain simulation models through increased re-use of simulation life cycle information at the domain level and at the design level over extended periods of time. The automated generation of executable analysis models from domain models will bridge the semantic gap between domain experts and simulation analysts.

Son et al. (2000) presented an approach proposed by National Institute of Standards (NIST) of USA to address the model building issue. The solution proposed by researchers at NIST

was the development of neutral libraries of simulation components and model templates. Components would contain detailed, formal, information models of all commonly used simulation components such as queues, machines and transporters. Each of these component models would have views tailored to specific modelling scenarios. Each model builder [translator] generates a model for the specific simulation package. The availability of such libraries together with the requisite translators, would simplify the model building process. It would also enable component based modelling, model reuse and internet based services, all of which could reduce the complexity and effort of simulation in manufacturing (See Figure 5.1).

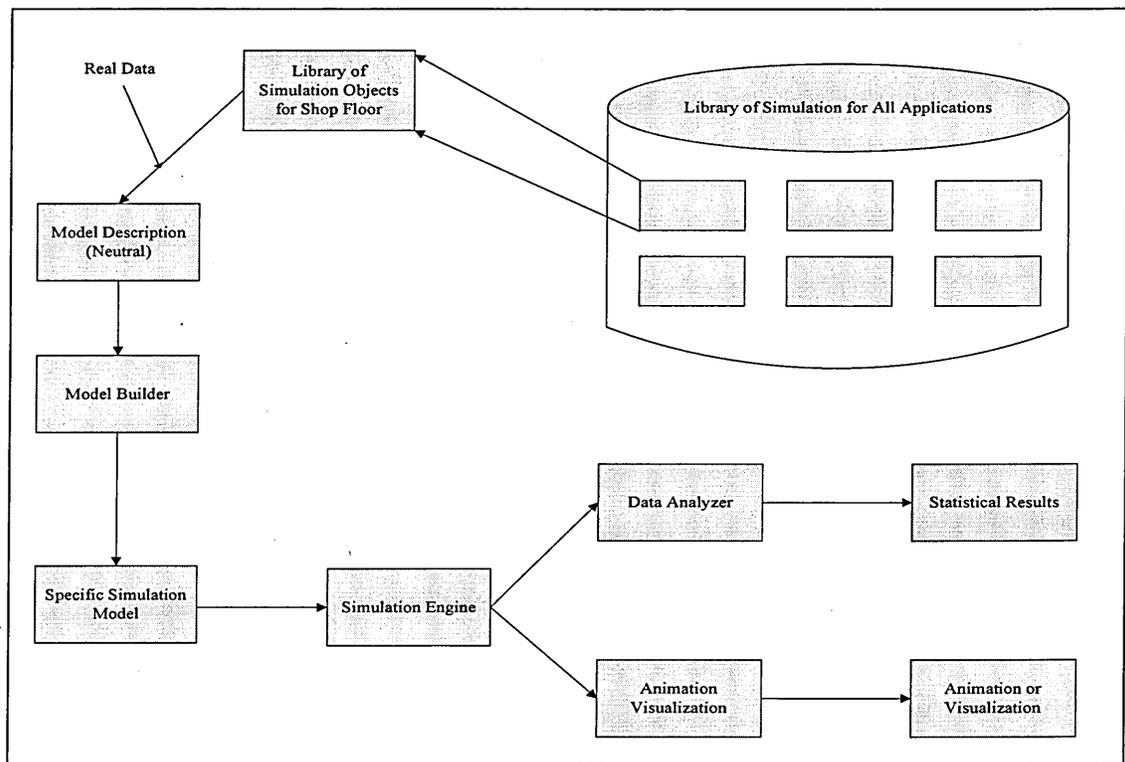


figure 5.1 : Neutral Library Approach to Develop Simulation Models (Son et al.,2000)

Weidemann (1999) recommended models are stored in a relational database. The data stored in the database include:

- Model data with all parameters and definitions of the model behaviour
- Experiment parameters and optimization methods
- Simulation results and statistical values.

Figure 5.2 shows the system architecture of the proposed approach.

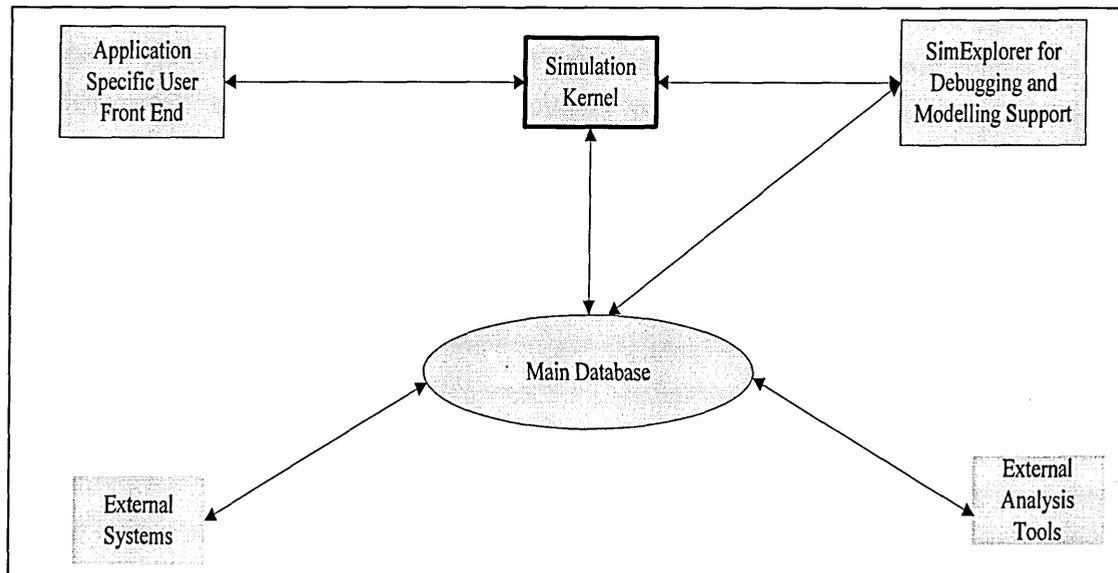


Figure 5.2: The System Architecture (Weidemann,1999)

Behaviour of objects and the behaviour of models are stored in the database by using micro-functions. Some examples of micro-functions are

- *Generate, Enter Request*
- *Store, StoreWarehouse, Queue etc.*

Then for example, a **Machine** object could be represented as EnterRequest, Enter, Operate, Sendto etc. The kernel mirrors the model from the database into the memory and executes the simulation. All simulation results are stored in a results table in the database.

5.2.1.2 Model Reuse: Library Driven Plus Knowledge Based Approach

Arons (1999) presented another reuse approach with the aim of reusing simulation models. According to that approach once a simulation model is developed, the parameterized model is stored in a database with the intention of reusing them later. When it is necessary to build a new model the modeller tries to retrieve an already existing model from the database similar to one that is to be developed. This is done by formulating a query on the database. Since the modeller in principle has no knowledge or does not want to have knowledge of the parameters used to describe the model, an expert system could be used to transform the design specifications in to the right query. However, it is very rarely that two models are matched exactly, nevertheless certain models exist which are very similar to the required model. It is up to the expert system to decide which model will be considered closest to the desired model.

Rathburn and Weinroth (1991) proposed the architecture shown in Figure 5.3 to accelerate the model development process. This approach facilitates managers to experiment in an inexpensive way, without having to be constantly assisted by a programmer, in a desktop environment. Beginning with a specific simulation model, a library of potential modifications to the attributes of the model is built. As the library took form, the major attributes to be considered by the user were identified and appropriate sections of code were written for each selection. This allows the manager to choose from a library of code representing the attributes relating to the problem at hand. The front-end expert system writes the simulation source code to an ASCII file based on the manger's answers to queries about desired attributes. Once the simulation model has been specified in this

manner, the manager runs the simulation. The manager can easily change the specification of any, or all, of the parameter attributes and run the simulation again.

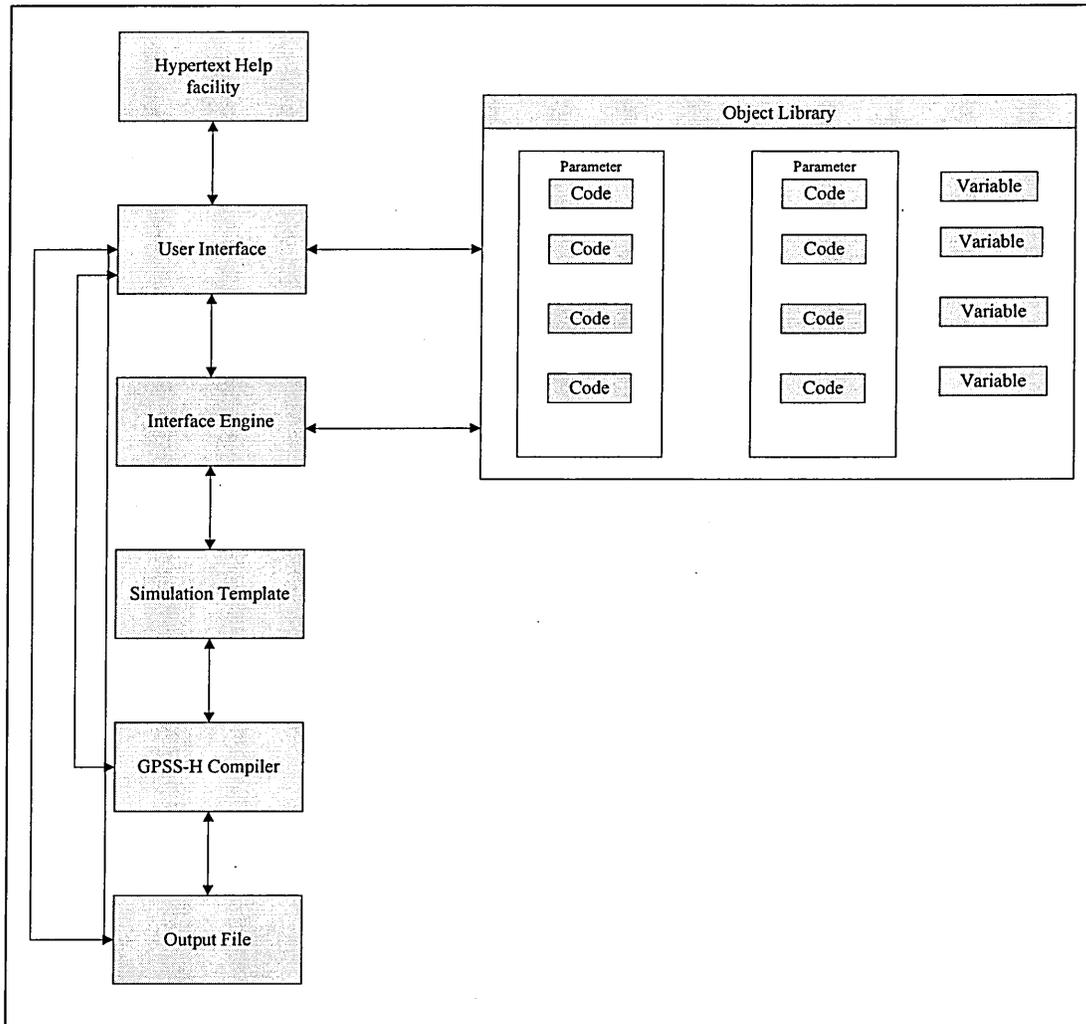


Figure 5.3: An approach to reduce necessity of a programmer in simulation model development (Rathburn and Weinroth, 1991)

The linking (between simulation software and expert system software) strategy follows the path of providing an intelligent front end which interfaces between the manager and the simulation, so that the services of the programmer are not needed in order to reconfigure the model for what-if experimentation. The authors have selected GPSS/H as the simulation

language and developed the expert systems in 1st Class Fusion-HT. Under this approach, different templates of codes are to be developed for each scenario of the problem.

5.2.1.3 Model Reuse: Object Oriented (OO) Approach

With the immense increase in research, development, and application of object oriented simulation (OOS) over the past decade, there is evidence that this technology is becoming the choice for modelling large, complex, and/or distributed systems. OOS is a system modelled and implemented object-oriented design and programming tools. According to Roberts and Dessouky (1998) the advantages frequently cited for using an object-oriented approach are the ability to model systems using entities that are natural to the system, faster development, increased quality, easier maintenance, enhanced modifiability, reusability of software and designs, evolvability and reduced development risks for complex systems.

The architecture of a system based on OO approach proposed by Praehofer (1996) is shown in Figure 5.4. This environment facilitates the representation of simulation application domains, the configuration of simulation models and the automatic generation of C++ simulation code from those configurations. Generic library components which can be parameterized are defined by C++ class definitions. An object-oriented database system stores and manages the object-oriented model of the domain, the different configurations already derived from it in the form of hierarchical model structures and the experimental conditions and the simulation results of the models. OO model supports the simulationist in the selection of a simulation model which represents a particular system design to be tested.

The environment supports

- the simulationist to run simulation experiments for existing model configurations and analyze the results

- the modeller to configure new systems designs based on the available domain knowledge
- the expert programmer to realize new model components and integrate them into the domain knowledge base.

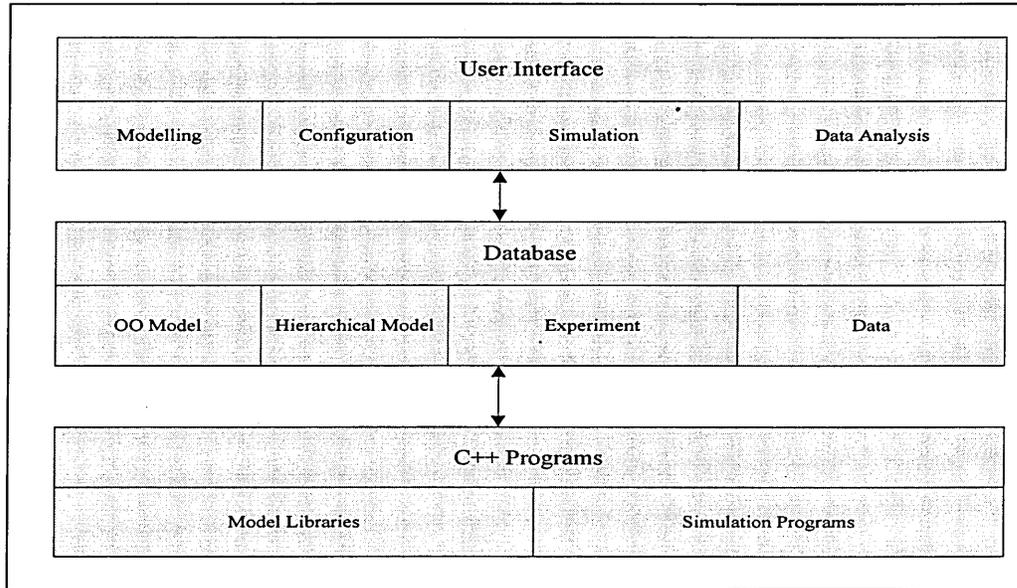


Figure 5.4 : Architecture of the domain modelling and configuration system (Praehofer, 1996)

Another OO approach, based on the application of software reuse and object-oriented methodologies for the modelling and control of flexible manufacturing systems, was proposed by Kovács et al. (1999). Some ideas of reuse in a hybrid (expert and traditional) simulation environment had been introduced by them to assist FMS design, implementation and evaluation. Components of FMS to be reused can be analyzed and defined using use-cases from Object Oriented Software Engineering (OOSE) and Rational Rose CASE tools. The reuse and the application of the object-oriented design techniques help to be able to build different FMS simulation models easier, faster and more reliably.

Anglani et al. (2002) proposed a new procedure to develop flexible manufacturing system (FMS) simulation models based on the UML analysis/design tools and on the ARENA simulation language. The two main features of the proposed procedure were the definition of a systematic conceptual procedure to design FMS simulation models and a set of rules for the conceptual model translation in a simulation language. The goal of the approach was to improve the software development efficiency through rule-based approach and to add some of the fundamental object oriented features to the ARENA simulation environment.

5.2.2 Integration with Other Packages

The researchers those who have adapted this approach had the following line of thinking. Although simulation has traditionally been applied to the design problem, it can also be used on an operational basis to generate production schedules for the factory floor (Miller and Pegden,2000; Koh et al.,1996; Rathburn and Weinroth,1991). The period allowed for planning a manufacturing schedule is much shorter as compared with typical gross simulation studies. Further, in a reasonably organized job shop, information required for simulation are available. But the real problem lies in the fact that most simulation languages and simulators do not readily interface with such data (Koh et al.,1996; Jain,1999).

The technique proposed by Seppanen (2000) can be used with a wide range of Visual Basic for Applications (VBA) supported tools such as Access, AutoCAD and Visio (See Figure 5.5).

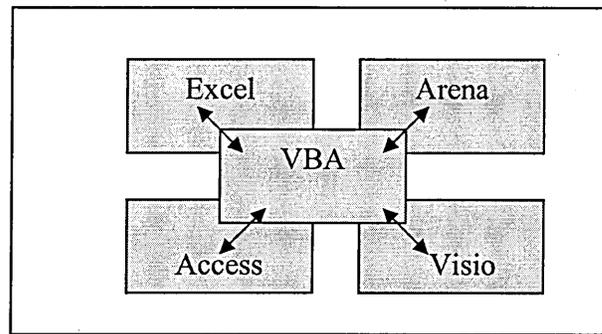


Figure 5.5 : Some Potential VBA Data Exchanges (Seppanen,2000).

This method allows simulation users to maintain and modify required model data without knowing the details of the simulation software. VBA code and its associated data are independent of the internal data structures of either Arena or Excel. Since VBA syntax is common for these applications, i.e. either Excel or Arena, it could be a part of Arena, Excel or written as a stand alone VB program.

Brown and Powers (2000) explained a technique called SIMFORCE to develop simulation models. Their approach is shown in Figure 5.6.

Under this approach a generic model (SIMFORCE Engine-An Arena model) had been developed for maintenance operations of US Air Force. Then the user inputs data through an Excel worksheet. The user defines the end item and its characteristics that affect processing (i.e. aircraft, its model, serial number and configuration), the processing steps it goes through and the resources required to support the processing steps and to fix the aircraft. When the simulation run ends, an Excel workbook which contains the output results collected during the simulation run is automatically opened. From the user's point of

view, the model is essentially a black box, hence “Simulation in a Box”. The user does not need to build a model or change the model logic to set up the model for his/her application.

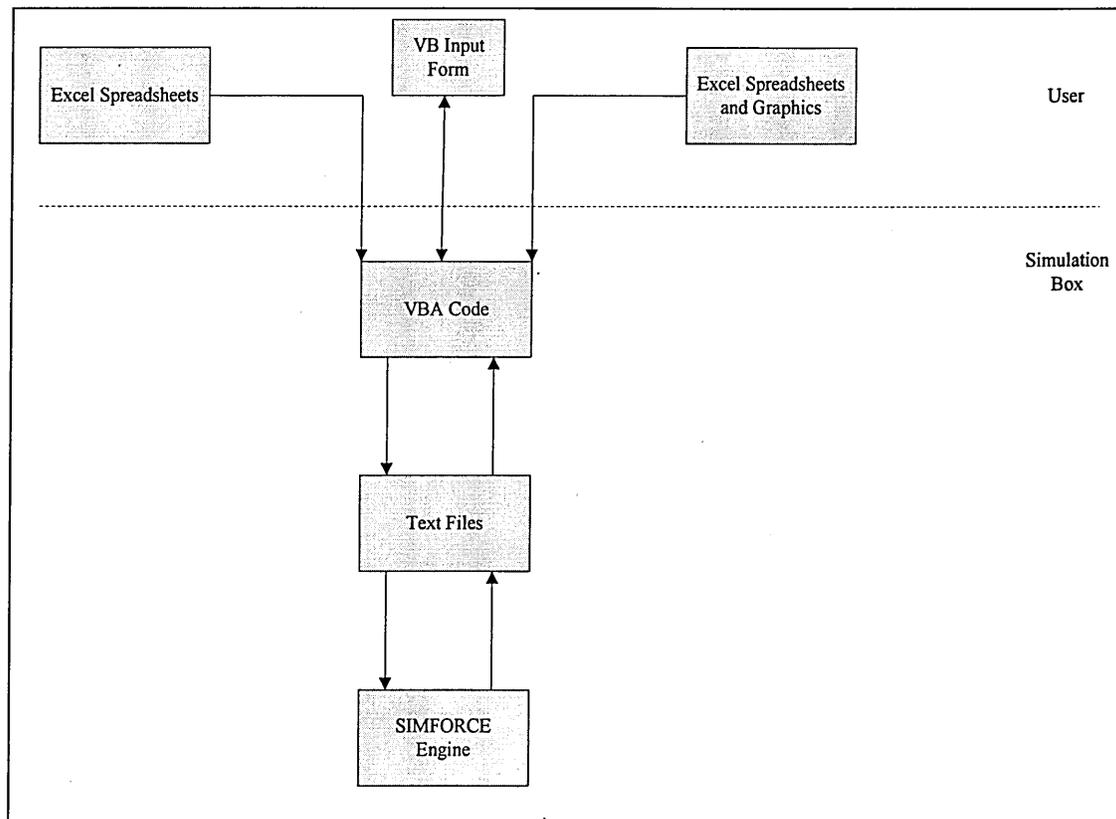


Figure 5.6: SIMFORCE Architecture (Brown and Powers, 2000).

Koh et al. (1996) has proposed a new approach for simulation-based scheduling, a subset of simulation, to answer the following main problems.

- a. Difficulties in obtaining accurate data for simulation model creation and subsequent updating, and simulation execution.
- b. Simulation modelling requires high level of expertise and time, especially when the model has to be constantly updated with the shop-floor changes.

Simulation modelling is essentially a translation process that translates the system definition from a particular data format into a specific format required by the simulation engine. If the simulation model format can be changed to adopt the same format as that of system data format, then the task of simulation modelling can be greatly eased. According to this approach simulation engine runs the simulations directly off the production database tables, a relational database, thus doing away with modelling of the system definition. In addition, any changes to the physical system that is reflected by the production database will automatically be updated in the simulation model without further need of data interfacing and translation. Since RDBMS and 4GLs are used, simulation modelling can be done without the analyst being too concerned with the programming language. Simulation models are reusable by adopting the proposed database model, with a common data structure.

Standridge (1999) proposed an approach called Object Manager Based approach. This is a simulation environment consisting of tools as well as the data needed for input to the tools and the data resulting from the use of the tools. Each tool as well as each input data set and output data set is considered to be an object. Each object may be characterized by attributes that can be given values. There are two basic classes:

- tools
 - simulation specific such as graphical model builder
 - entire simulation environment
 - general purpose software such as spreadsheet or a graphic package
- data sets
 - result from operation of tools
 - are constructed as input to tools

The modular simulation environment provides the structures for the flow of data between heterogeneous commercial software products and also it can link simulation specific software and general-purpose software. It supports the joint use of simulation and other analysis techniques such as optimization. It contains its own tools for managing data. Each tool and data in the environment is viewed as an object under the control of the object manager. The object manager organizes data sets and invokes the tools as needed, either independently or from within other tools. It ensures that the inputs to tools and the results from tools are complete and acceptable as well as integrating them into the environment.

5.2.3 Knowledge Based/Artificial Intelligence Approaches

A review in the field of knowledge based simulation systems (KBSS) has been presented by Merkuryeva and Merkuryev (1994). According to them, the development of KBSS results from combination of simulation with Expert System (ES) called knowledge engineering. The goal of KBSS is to imbed as much of the required knowledge and experience as possible within the software. Building and application of such systems will make it possible for engineers, scientists and managers to perform simulation studies correctly and easily without special training in programming languages, simulation theory and other skills required for simulation project realization. KBSS can be used in modelling, designing and running experiments and analysis stages of the simulation model development life cycle. Tools available for the development of KBSS include conventional computer languages, object-oriented languages, artificial intelligence (AI) languages, special purpose AI hardware, expert system shells, ES development environments and simulation development environments. The benefits expected from incorporating knowledge engineering include improved representational capability, shortened model

development time, reduced skill level requirements, faster model execution and improved system maintainability. A few attempts made to accelerate the simulation model development process based on knowledge based approach are found in the existing literature.

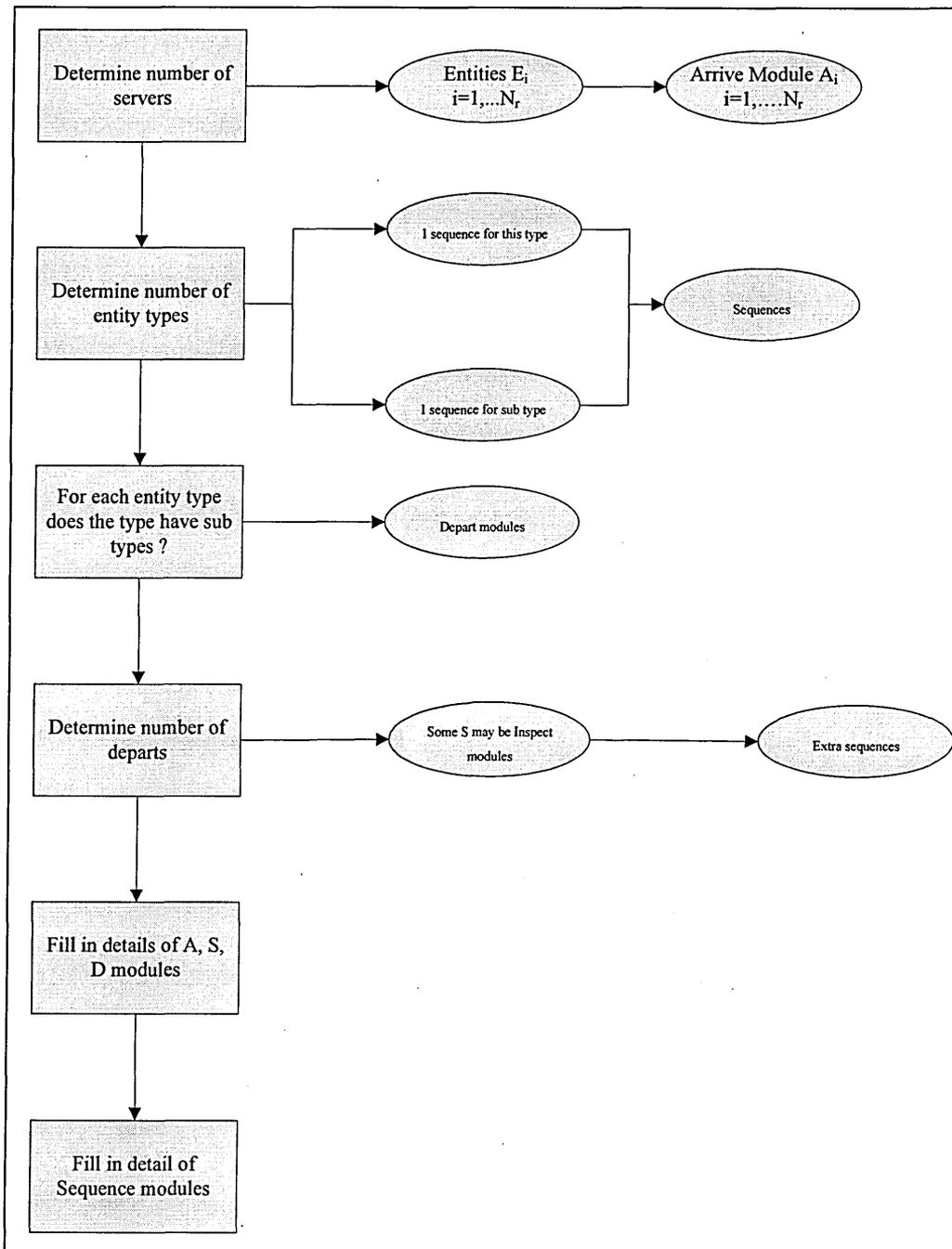


Figure 5.7: Design Schema (Arons and Asperen, 2000)

Arons and Asperen (2000) proposed to use a knowledge-based system essentially to support the modeller, who may have extensive domain knowledge but little experience with simulation tools, in the decision making process. Figure 5.7 shows schema for designing the model with only Arrive, Server and Departure modules. This approach is formalized and programmed by using a programming language or a knowledge based tool. Information provided by the modeller (Answers for number of questions) are the inputs for such a program. The level of output may vary from providing list of suggestions with respect to the number of modules to be used, the values of parameters etc. to the automatic generation of the simulation model in the selected simulator. In this case researchers have attempted to use the proposed approach with Arena.

According to Pitts and Hwang (1999), simulation packages currently available were all developed with the professional modeller in mind. In order to extend the world of simulation modelling to casual computer user they presented a methodology of an intelligent interface and use of an Intelligent Agent, called SMIART. An Intelligent Agent (IA) is a computerized module that simulates the reasoning process of a human, imprecise yet potentially essential for decision systems in real-time complex environments. A set of prompts is presented to the user through user interface and specific information is asked about the model desired. Interface is capable of querying the **Intelligent Agent** for information it needs. Once all the information has been acquired either from the user or generated by the IA, SMIART then passes all information to the simulation engine, which handles the actual generation of the model (See Figure 5.8).

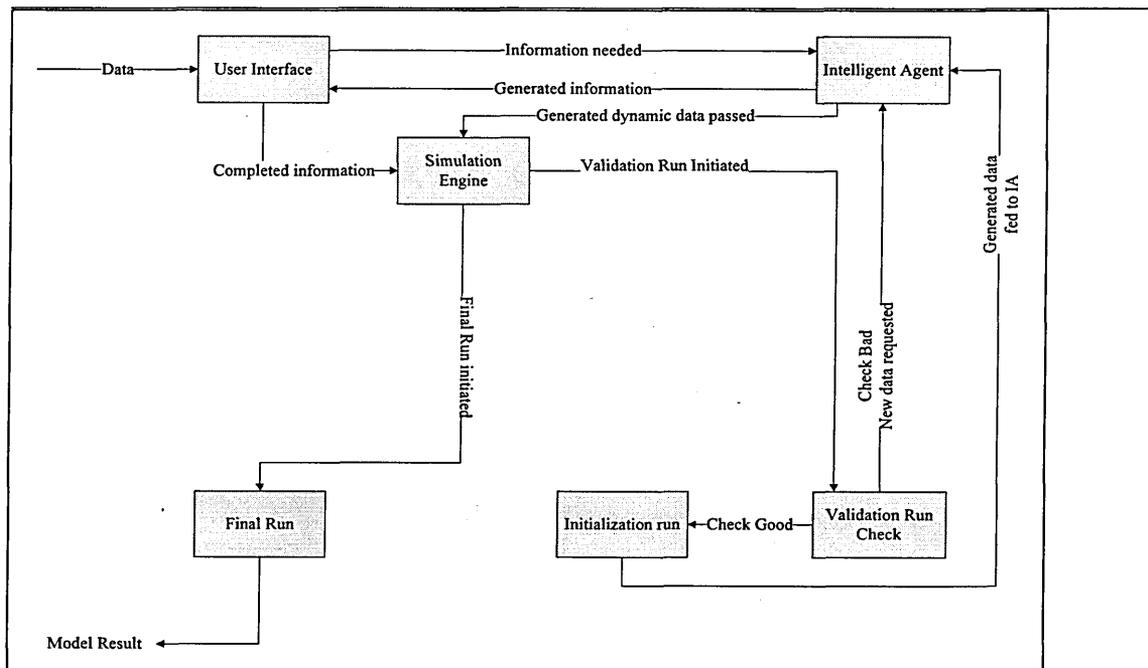


Figure 5.8: Architecture of SMIART (Pitts and Hwang, 1999)

Chun (1997) presented a paper to describe research and development in simulation program synthesis using a knowledge-based approach for airport check-in counter allocation. The system, SPEEDCHECK Profile Simulator, was encoded with all the domain expertise and simulation knowledge needed to automatically generate, run and animate simulation programs related to airport checking counter allocation. A domain dependent expert system using rule based reasoning technology has been developed as a front-end to the SIMAN simulation package (See Figure 5.9). The Expert System Module contains a set of rules or heuristics that captures subtle variations in the airport statistics based upon destinations, airline and time of the day. The main objective of that system was to make simulation easily accessible to non-technical management-level people who might not have a background in computer programming or simulation theory.

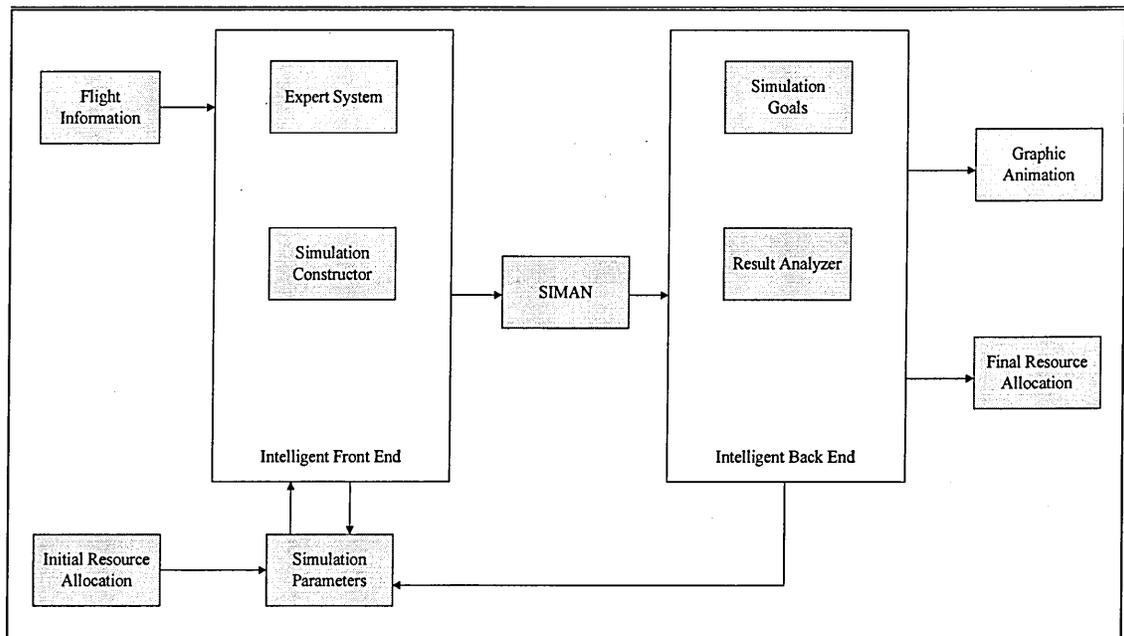


Fig 5.9: Components of SPEEDCHCK Profile Simulator used in the resource prediction mode (Chun, 1997).

Sakthivel and Agarwal (1992) proposed a knowledge-based approach for constructing a simulation of an information system. There were two steps in the overall methodology;

1. Knowledge pertaining to the target information system gathered by an analyst during requirements analysis is synthesized into a PetriNet model of the IS.
2. This model is then utilized for automatically building the IS simulation model in GPSS.

The graphical Petri Net model forms the template for the construction of the knowledge base. There were four kinds of knowledge.

- Knowledge related to the Petri Net model of the IS

- Knowledge related to the mapping between the Petri Net representation and GPSS blocks
- Knowledge required to combine these blocks in to a complete and correct GPSS specification
- General purpose or house keeping knowledge related to maintaining the knowledge base

The Petri Net diagram was represented by using an incidence matrix, which shows the relationship between places and transitions. The assertions were obtained automatically from the incidence matrix description of the Petri Net. This would identify a list of

- All places
- All transitions
- All places that are outputs of at least one transition
- All places that are inputs to at least one transition

Knowledge base also contains several rules that operate on these assertions and identify needed GPSS simulation blocks. According to Sakhivel and Agarwal (1992) the benefits of this method were

- Shortening the system development cycle
- Economical and fast development of simulation model from the specifications obtained during requirements analysis
- Allowing analysts with little training in simulation to test the IS configuration for user specified performance requirements
- Flexibility in modifying system configurations

In order to reduce [simulation model] development time, the amount of knowledge regarding simulation methodology and programming language, and also to increase program standardization, an automatic code generation system based on Petri Nets(PN) was developed by So and Lew (1999). The stages of this approach are

- Construction of a PN model by the user following Simulation-Based Petri Net (SBPN) labelling conventions, a convention developed by authors. Representation of the SBPN model using a tabular internal computer data structure that facilitates the code generation process. The SBPN would then be represented by using a decision table, using systematic translation rules, which can be done manually or by a GUI.
- Actual code generation for a specific target language (in this case GPSS). Tabularly represented PN is the input data file to the code generator and the output is the target program code for simulating the IS.

Under this approach no debugging of the *low-level* target programme was required by the simulationist. Any changes would be made to the *high-level* SBPN model with the subsequent phases repeated.

Another Knowledge-based simulation modelling approach was presented by Murray and Sheppard (1988). According to that approach both the acquisition and programming tasks, performed by a simulationist in the traditional approach shown in Figure 5.10 can be automated.

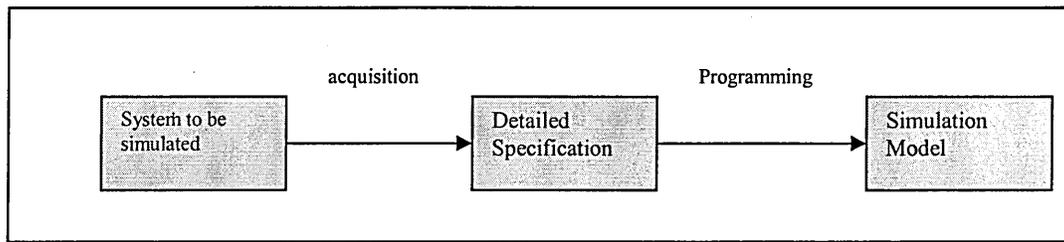


Figure 5.10: Traditional approach to simulation model construction (Murray and Sheppard, 1988).

There were three approaches to automate specification acquisition portion of traditional simulation model construction; Natural Language Processor, Graphics Interface and Dialog Monitor (See Figure 5.11).

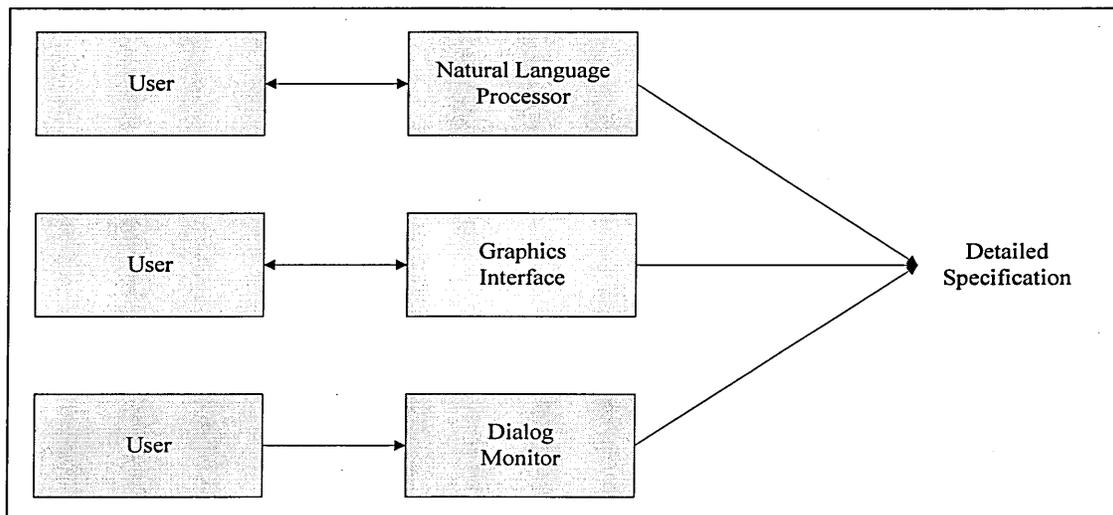


Figure 5.11 : Automated Approach to specification acquisition (Murray and Sheppard,1988).

A detailed description of the system may be obtained from English language descriptions via Natural Language (NL) Processor or through interactive dialog monitor. A graphics interface was most useful for describing the structure or layout of components within the system. Once the pertinent detailed information or model specification had been obtained the model can also be automatically constructed, using either a program generator or an automatic programming system (See Figure 5.12).

In this approach only the dialog monitor is used to obtain system specifications due to practical difficulties. KBMC consists of three kinds of knowledge; domain knowledge i.e. knowledge of a subset of queuing systems, general simulation modelling knowledge and target language knowledge i.e. in this case SIMAN knowledge. So

Domain knowledge + modelling knowledge → extraction rules → Guide interactive specification session with the user

Modelling knowledge + target language knowledge → construction rules → transform internal specification into executable model in the target language.

The model construction and formatting portion of the KBMC system was completely transparent to the user. A user who is familiar with the system to be simulated, but is not familiar with the target language, can construct executable models.

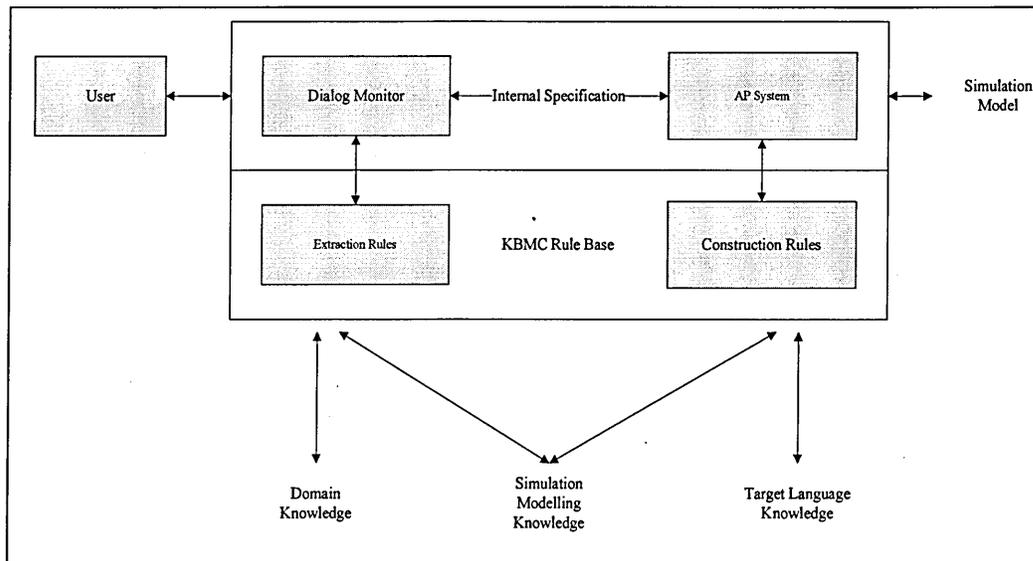


Figure 5.12: KBMC System Overview (Murray and Sheppard, 1988).

It is found that various researchers have tried different approaches to accelerate the simulation model development process as they have thought that their approach is the best way to accelerate the simulation model development process. In the next section the advantages and disadvantages of each of the identified approaches are discussed in order to answer the next research question i.e.: **Is there a need for a new methodology for accelerating the simulation model development process? If so, what is it?**

5.3 Review of Existing Rapid Simulation Model Development Approaches

A number of different approaches for accelerating the simulation model development process were described earlier. The identified approaches were model reuse: library driven approach, model reuse: library driven plus knowledge based approach, model reuse: object oriented approach, integration with other packages and knowledge based/artificial intelligence approach. These approaches are critically reviewed below.

5.3.1 Model Reuse

The basic premise behind model reuse is to store complete models (Weidemann,1999; Arons,1999), components of models (Mertines et al.,2000; Rathburn and Weinorth,1991) or combination of complete models and components (Benjamin et al.,2000; Son et al.,2000) in a library (database) for later use. When there is a need to develop a new model, the required model is developed by either extracting a complete similar model from the library or by combining components selected from the library. According to Pressman (2000) many software practitioners continue to believe that reuse is “more trouble than its worth”. Paul

and Taylor (2002) pointed out that in the world of COTS [commercial-off-the-shelf] simulation packages, it is difficult to see practically how one can trust a model without detailed verification that may be more costly than developing the model from the start. Therefore, Pressman's expression is valid in the case of simulation model reuse too. Theoretically, this library approach sounds well. But in practice, hardly two models are matched with each other. Therefore it is very difficult, if not impossible, to find a similar model from a library either manually by the modeller or by the computer with the help of a knowledge based system¹.

Even if it is assumed that the library driven approach will become successful in simulation model development, there are some problems with this approach. Firstly, a large number of models are to be built and stored in a library to represent various systems. For example a manufacturing system may have large number of possible alternative models because there is nothing called 'the manufacturing system'. But building a library with large number of models will be a very expensive and time consuming task. Secondly, if a model is developed with the intention of using it later, it must be developed systematically and the model must be well documented. Therefore, the cost and time of building the original model would be higher when compared with a model developed to 'use and throw away'. This will substantially delay the development of the original model. Thirdly, even when a model or components are selected from a library, the modeller would have to spend a substantial amount of time understanding the model, logic, variables etc. before it is used or any alterations are made. This may nullify the time saving made by using the library approach.

¹ Results of the questionnaire survey which are explained in chapter six also revealed that the respondents disagree with the statement that the previously built models can be re-used to build new models.

5.3.2 Library Driven Plus Expert System Approach

The second identified approach, library driven plus expert system approach is an extension of the library approach. The basic idea behind this approach is to extract models from a library (database) by getting answers to a set of questions (queries) from the user. The examples of queries may be the number of machines, the number of arrival points, the number of departures etc. However, building a model based on answers to a set of questions alone is not possible because simulation is not only a science but an art as well. The programming and statistical components are the science part and the analysis and modelling components are the art (Shannon,1998; Banks,2000). Therefore, the major problem with this approach, on top of the issues mentioned earlier under the library driven approach, will be embedding all the knowledge required to build a model into the expert system.

5.3.3 Object Oriented Approach

The basic idea behind the objected oriented approach is to build reusable classes. However, to build a model using the OO approach, first it should be translated into the required format. In addition to that the modeller should think in terms of O-O paradigm, which may not be familiar to him. Another problem with this approach is that most of the available popular simulators/simulation languages are not object oriented. Therefore the modeller will have to use an object-oriented language such as C++ or a lesser known O-O simulation language such as eM-Plant in developing the model. This will make the modeller's task

difficult and prevent him/her from using facilities provided by popular simulators. These factors may substantially reduce the time saving gained by this approach.

5.3.4 Integration with Other Software

The next identified approach was accelerating the model development process by integrating software such as spreadsheets, databases and drawing packages with the simulation software. This approach can be used only with a specific model, i.e. for example maintenance model as proposed by Brown and Powers (2000), and is not an approach to accelerate the generic model development process. Further, quite lot of programming, may be needed using a language such as Visual Basic [for Applications], to integrate those different software such as MS_Excel, MS_Access and Visio with the simulation software. Most of the present simulators are capable of integrating data from different software. For example, Arena has facilities to import and export data to and from databases/spreadsheets and is also capable of integrating visual basic modules with simulation models. The users can make a limited set of pre-defined alterations to an already developed model by changing parameters through a spreadsheet or a database. Therefore this approach may be considered useful in accelerating the experimentation process rather than model development process.

5.3.5 Knowledge Based/Artificial Intelligence Approach

The last identified approach is knowledge based/artificial intelligence based approach. One of the main objectives of this approach was to allow users having good domain knowledge

but little experience with simulation software to develop models without having help from a modeller. Basically the expert system/knowledge based system develops the model automatically by obtaining answers for a set of queries from the user. But to build a model to represent a complex system a lot of cross-questioning may be needed. Programming this human reasoning process is not an easy task. This approach may be successful for a specific domain dependent application. For example airport checking counter system (Chun,1997), which is almost the same for any airport in the world and IS system (Sakthivel and Agarval,1992). But for a system such as a manufacturing system, which is different from one factory to another, embedding the knowledge required for modelling to the expert system may not be an easy task because it is impossible to identify a generic manufacturing system.

5.4 Is there a Need for a New Methodology?

Even with all those mentioned issues with the identified approaches, they may still accelerate the simulation model development process. However, all these researchers have tried to improve the efficiency of the model development process. According to Drucker (1992) effectiveness is the extent to which desired result is realized while efficiency is output divided by input, or the extent to which the result produced was produced at least cost. Drucker (1992) pointed out efficiency is concerned with doing things right and effectiveness is doing the right things. In the area of management, where these two concepts are frequently used, effectiveness is setting the correct objectives and achieving them while efficiency is obtaining the maximum output with minimum amount of resources (inputs). In the context of simulation, effectiveness is to build the right model to represent

the actual system and efficiency is building the model in the most efficient way, i.e. in a short period of time and at a less cost. From the earlier analysis of the existing approaches of the rapid simulation model development techniques it is clear that those researchers have paid more attention to improve the efficiency of the model development process. Whether it is model reuse, integration with other software or knowledge based expert systems, the objective is to reduce the time by reducing the programming burden. Simulation model development process may be accelerated with the previously mentioned approaches, however there is no guarantee that the model developed in a shorter time is the right model that represents the actual physical system. If it is found that the developed model is not the model required by the user at the validation stage again the changes are to be made to the computer model. This may nullify the saving achieved through the programming efficiency. There is not much evidence found in the contemporary literature on improving the effectiveness of the simulation model development process. i.e. develop the right model first and thereby reduce the time required to develop the model. Therefore there is still a scope for a new methodology for accelerating the simulation model development process by improving the effectiveness of the process while applying the most efficient programming techniques. Therefore the answer to the next research question, i.e. **Is there a need for a new methodology for accelerating the simulation model development process?** is yes.

5.5 Summary

In this chapter it was found that two major reasons for lengthening the simulation model development process are the lack of understanding of the system to be simulated between

the user and the modeller and the difficulty of programming. However, it was found that the existing approaches to accelerate the simulation model development process are more directed towards reducing the programming task of the programmer. They do not provide much support to improve the understanding of the system to be simulated. Therefore, there is still a need for a new methodology for accelerating the simulation model development process. The other research strategy which was identified in the third chapter was a questionnaire survey. The results of the questionnaire survey conducted amongst simulation practitioners and academics are explained in the following chapter.

Chapter Six

Results of the Questionnaire Survey

As shown in the third chapter, a questionnaire survey was one of the strategies of the research methodology. The objectives of the questionnaire survey were to validate (accept or reject) the assumptions made on simulation model development process based on the literature survey and to develop the background for a new approach on the rapid simulation model development process. The sample, questionnaire and results of the questionnaire survey are presented in this chapter.

6.1 Questionnaire

As mentioned in the second chapter, current literature suggests that the model development phase is the most time consuming phase among the stages in the simulation project life cycle. Therefore, there is a need for simulation software which provides efficient and simple methods which may lead to acceleration of the simulation model development process. To justify this belief and to find a direction for a new approach for accelerating simulation model development process the following assumptions were made based on the literature which was presented in the second and fifth chapters.

- Model building is the most time consuming phase of the simulation project life cycle.
- A conceptual model is built before the computer model is built.
- There is a tendency to undertake simulation projects by persons having non-engineering/mathematics background.
- Previously built models can be re-used to build new models

- Simulationists use simulation languages and general purpose languages instead of or in conjunction with simulators.

However, since all the mentioned assumptions were solely based on the literature, an e-mail survey was conducted amongst academics and practitioners in the simulation area in order to verify the findings of the literature survey and ensure that they are still applicable. Since e-mail addresses were widely available in mail lists, e-mail mode rather than conventional postal mode was selected for the questionnaire survey to reach a large audience representing a wider geographical area¹.

A nine-question questionnaire (see Annexure I) was designed to collect information from respondents regarding the current profession, degree/professional qualifications held, number of simulation models built, their perception about the difficulty of each stage of the simulation project life cycle, percentage time required to complete each stage of the simulation project life cycle, types of software used to develop simulation models (Special Purpose Simulation Languages/Simulators/General Purpose Language) and type of other software used to model development/experimentation (MS_Excel, MS_Access, Visio, VBA). One more question was included to measure the agreement with certain statements made in the literature regarding the simulation model development. The questionnaire was kept as short as possible by including only the essential questions in order to encourage the participants to respond and thereby to increase the response rate.

¹ Advantages of using e-mail over the post in questionnaire survey were explained in chapter three

6.2 Sample

This questionnaire was distributed to electronic mail boxes of the members of the SIGSIM simulation group (520 members), one of the largest simulation groups sponsored by Association for Computing Machinery (ACM) and the participants of the winter simulation conference (377 e-mail addresses), one of the largest conferences of the world on discrete event simulation. Fifty-four of the recipients returned filled questionnaire. Findings of the questionnaire survey are given in the next section.

6.3 Findings

The objective of the first three questions was to establish the background of the respondents. Table 6.1 shows the percentage of respondents having different qualifications. The majority of the respondents have a qualification/degree in operations research area (Sum is not equal to 54 because some of the participants had more than one qualification).

Qualification	Total	% (out of 54)
Production Manufacturing Engineering	7	12.96
Mechanical Engineering	3	5.56
Operations Research	27	50.00
Systems Modelling	9	16.67
Software Engineering	13	24.07
Business Management	9	16.67
Other (Chemical Eng)	1	1.85

Table 6.1: Qualification/Degree held by respondents

The experience of the respondents in the simulation area is shown in table 6.2. Even though it is not conclusive, it seems that simulation lost its popularity during the late eighties and early nineties and gained it back again after mid nineties. The fact that 81% of the respondents have at least 5 years of experience in the field of simulation should provide a degree of credibility to the findings of the survey.

Experience	Total	% (Out of 54)
Under 5	10	18.52
5-10 Yrs	11	20.37
10-15 Yrs	6	11.11
15-20 Yrs	12	22.22
Over 20	15	27.78
Total	54	

Table 6.2: Duration of involvement of respondents in the simulation area

The focus of the rest of the questions was on the simulation model development process. The objective of the next question was to measure the difficulty of different tasks to be completed during a simulation project. Respondents were asked to indicate the perceived difficulty of each stage on a scale from 1 to 5 (Most difficult-1 to Least difficult -5). The results revealed that conceptual model development is the most difficult task while experimentation is the least difficult task.

Stage	Difficulty	Rank
Problem/Objective Definition	2.87	3
Conceptual Model Building	2.51	1
Simulation Model Development	2.74	2
Experimentation	3.6	5
Project Completion and implementation	3.19	4

Table 6.3: Weighted average difficulty of each stage of the simulation project

Table 6.4 shows the average proportion of time needed for each stage of the simulation project life cycle. This table shows that model building and testing is still the most time consuming task of a simulation project.

Stage	%	Rank
Problem/Objective definition	18.8	4
Model building and testing	40.7	1
Experimentation	25.3	2
Project completion and Implementation	20.8	3

Table 6.4: Average proportion of time needed for different stages of the life cycle (%)

Figures 6.1 and 6.2 compare the results of the questionnaire survey with the facts found in the literature. It shows that even with all the developments in simulation software in last 10 years model development continues to be the most time consuming task of the simulation project life cycle.

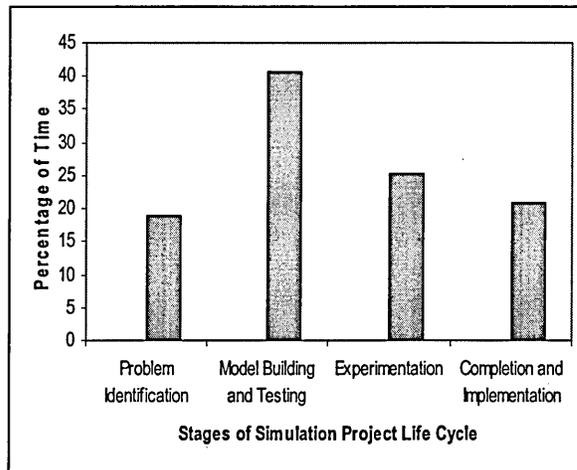


Figure 6.1: Percentage time needed for each stage of a simulation project – Questionnaire Survey 2003

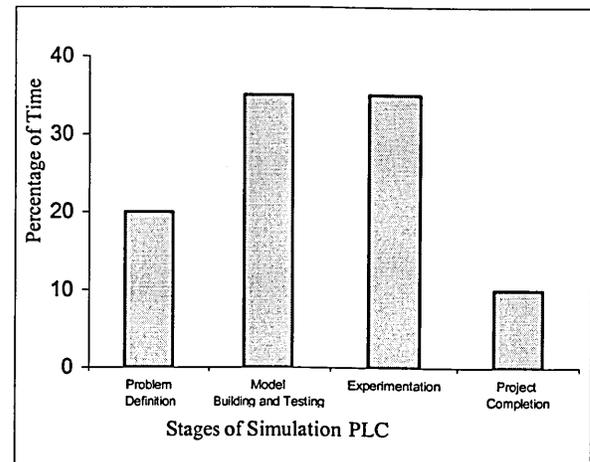


Figure 6.2: Project time –scales : a percentage breakdown(excluding implementation) (Robinson, 1994)

Tales 6.5, 6.6 and 6.7 show the percentage of respondents using different simulation languages, simulators and general-purpose languages for model development respectively.

Simulation Languages	Number	As a % of total
SIMAN	11	20.37
SLAM	6	11.11
SIMSCRIPT	3	5.56
GPSS	5	9.26
Other	5	9.26

Table 6.5: Percentage of respondents use different simulation languages

As shown in table 6.6, Arena is the most popular simulator.

Simulators	Number	% of total
Arena	22	40.74
Extend	7	12.96
Promodel	7	12.96
Automod	3	5.56
Witness	3	5.56
Simul8	3	5.56
AweSim	3	5.56
Emplant	3	5.56
Quest	1	1.85

Table 6.6: Percentage of respondents use different simulators

Table 6.7 reveals that general purpose languages are widely used for simulation model development. Nine of the respondents (17%) use solely a programming language (neither simulation language nor simulator) for simulation model development process.

Language	Number	As a % of Total
C++	26	48.15
VB	13	24.07
Java	9	16.67
Other	9	16.67

Table 6.7: Percentage of respondents use general purpose programming languages

The number of respondents use spreadsheet, database or drawing software and VBA to integrate them in simulation model development/experimentation, as a percentage, is shown in table 6.8.

Package	Number	As a % of Total
MS_Excel	39	72.22
Access	15	27.78
VBA	13	24.07
Visio	9	16.67

Table 6.8: Percentage of respondents use spreadsheet/database/drawing and integration software.

The objective of the final question was to know the level of agreement regarding few statements formed based on the contemporary literature on literature. Agreement varied from strongly agreed to strongly disagreed and was measured in a scale from 1 (strongly agree) to 5 (strongly disagree). Table 6.6 shows the results.

Statement	Agreement
Models are frequently simplified due to limited time allocated for projects	2.5
Sometimes I had to reduce the time allocated for experimentation because I had spent more time on model building	3
Previously built models are frequently re-used to build new models	3
There are a large number of people who have not studied a scientific discipline, building simulation models	3
New issues arose during the experimentation stage which led me to build new models/modify built models	2
I build a conceptual model before building a simulation model	2

Table 6.8: Agreement with statements (Strongly agree –1 to strongly disagree –5)

6.4 Discussion

As a result of the literature survey, several assumptions were made. These were shown in section 6.1. The following section discusses whether the findings of the questionnaire survey confirm or nullify these assumptions.

- Model building is the most time consuming phase of the simulation project life cycle.

As shown in table 6.4, on average 41% of the total project life cycle time spend on model development and testing. Further, according to table 6.3 the most difficult tasks of the simulation model development process are conceptual model development and simulation model development which are sub-stages of the model building process. Therefore the findings of the survey support this assertion. This finding provides empirical evidence to justify the formulation of the research question which was done in the third chapter.

- A conceptual model is built before the computer model is built.

In general the respondents agree with this statement. Further, according to the responses conceptual model development is the most difficult task of a simulation project.

- There is a tendency to undertake simulation projects by persons having non-engineering/ mathematical background as well.

Generally the respondents do not agree with this assumption. The constitution of the respondents also does not support this idea. Only 2% of the respondents have no qualification in a science/mathematics discipline. Further, only 3% of the respondents use

solely a simulator for simulation model development, i.e. 97% of the respondents use at least one simulation language or a general purpose programming language. This indicates that even with all efforts made by simulation vendors to make their products more user friendly, simulationists still find it difficult to build models using only simulators. This situation may discourage persons without a qualification in a scientific discipline to enter into the simulation field.

- Previously built models can be re-used to build new models

Even though this assumption is made with a positive connotation towards the re-use of simulation models, literature are found both for and against this idea. However, the findings of the survey do not confirm this assumption. This finding supports the claim made regarding the re-use of model in simulation, i.e. re-use is theoretically promising but practically difficult to implement approach, in chapter five.

- Still simulationists use simulation languages and general purpose languages instead of or in conjunction with simulators.

There is strong evidence to support this assumption. As mentioned earlier 97% of the respondents use at least one simulation language or a general purpose programming language. Further 9% of the respondents use only a general purpose language for simulation model development. This indicates that even with all the improvements in simulators, modellers still require the flexibility provided through programming languages for developing simulation models. However, the problem with using a programming

language for simulation model development is that the modeller has to spend lot of time to learn a programming language. Hence, there is still a need for improving model development capabilities of simulators. This will allow the users to develop simulation models by using simulators rather than learning a new language. Another finding of the survey was that seventy three percent of the respondents use at least one of the following packages; MS_Excel, MS_Access or Visio, either in the model development or experimentation stages.

6.5 Summary

The findings of the questionnaire survey provided empirical evidence supporting the propositions, i.e. model development is the most time consuming phase of a simulation project and re-use of models in simulation is a difficult task, as expressed in previous chapters. Other important findings of the survey are that simulation is still a field dominated by people having a scientific background, even with all the developments in the simulation software, simulationists frequently use other programming languages and packages and conceptual model development is the most difficult task of a simulation project.

Simulation model is also a piece of software. Therefore, before presenting a new framework for accelerating the simulation model development process based on the findings from case study research, literature review and questionnaire survey, different software engineering methodologies are discussed in the next chapter.

Chapter Seven

Software Development Methodologies

According to Sommerville (2001), software is computer programmes and associated documentation. In that sense simulation model is also a piece of software. For example, it is similar to development of payroll or stock control software for a business organization. In that case, a computer programmer develops the software package by using a computer language such as Visual Basic or C. Where as in simulation, a modeller develops the model of the system to be simulated by using a simulation software such as Arena or Extend. In both cases, programmer or the model developer develops the final product, i.e. payroll package or the simulation model, by gathering information from the client.

According to (Sommerville, 2001), there are four fundamental process activities common to all software development processes.

1. *Software specification* -the functionality of the software and constraints on its operation must be defined.
2. *Software Development*- The software to meet the specification must be produced
3. *Software validation* - The software must be validated to ensure that it does what customer wants
4. *Software evolution* - The software must evolve to meet changing customer needs."

These four stages are analogues to the stages of simulation model development process which were discussed in Chapter two. Therefore, simulation model development process is kind of a software development process. As such, it is worthwhile to analyse different

methodologies used to accelerate the general software development process with the view of adapting them in accelerating simulation model development process.

The objective of this chapter is to identify different rapid software development methodologies and techniques. However, before it goes to explain rapid software development approaches, traditional approach of software development, which the weaknesses of that approach led to the development of new approaches to accelerate the software development process, is discussed. This chapter concludes with an examination of different tools used under the rapid development approaches.

7.1 'Waterfall' or Linear Sequential Model

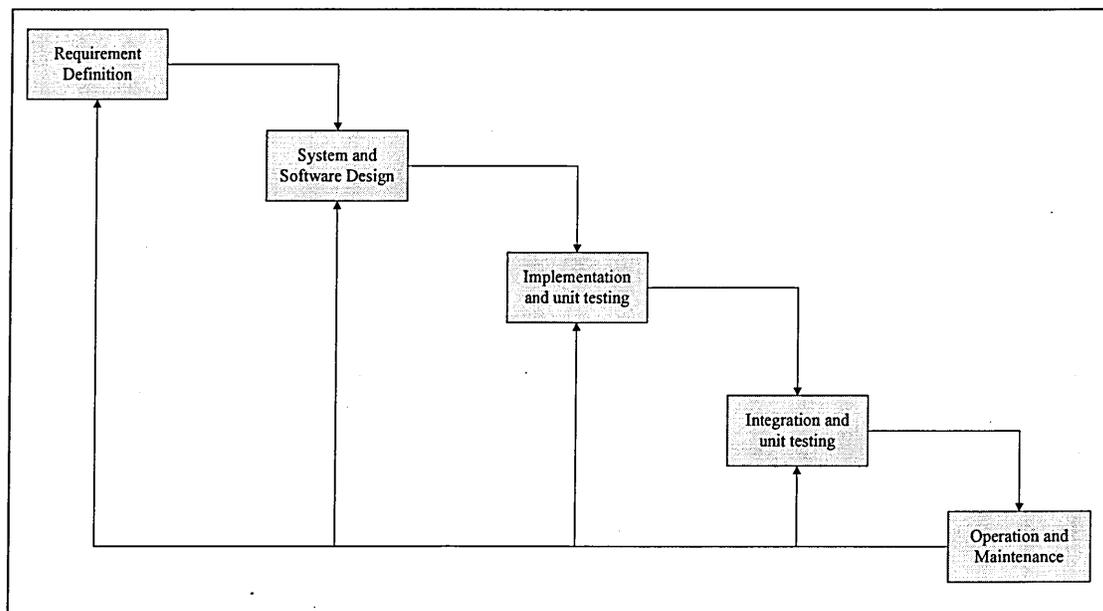


Figure 7.1 : The software life cycle (Sommerville,2001)

The original version of the waterfall model (Sometimes called as "classic life cycle" or "Linear sequential model" [Pressman (2000)]) was first illustrated by W. W. Royce in 1970 (Pressman,2000; Sommerville,2001; Carr,1989). According to Pressman (2000), the linear sequential model is the oldest and widely used paradigm for software development. This approach, as presented by Sommeville (2001), is shown in Figure 7.1.

Under this approach software development is done in stages. According to Howard (1997), under the traditional approach (with the evolution of other approaches, which are discussed in the latter part of this chapter, some authors have identified the waterfall approach as the traditional approach as well) requirements are explored and analysed before a system to meet them is designed. If accepted, the design is implemented and tested. Eventually, when users, clients and developers are satisfied, the system is made operational. According to Pressman (2000), in principle, the result of each phase is one or more documents which are approved ("signed off"). The following phase should not start until the previous phase has finished...it is normal to freeze part of the development, such as specification, and to continue with later development stages. Therefore, the four basic stages of waterfall approach are design, code, test and delivery/support (Carr, 1989; Pressman, 2000).

7.1.1 Issues in Adapting Waterfall Model

According to Sommerville (2001), the problem with the waterfall model is its inflexible partitioning the project into distinct stages. As such it is difficult to respond to changing customer requirements. Therefore the waterfall model should only be used when the requirements are well understood. Verner and Cepra (1997) state that the waterfall method is a phased approach in which each phase should be completed before the next is started. A

clear statement of requirements is necessary before software design and implementation can start. However, it is often difficult to obtain a complete set of requirements. As per Howard (2000), requirement determination is plagued by communication problems. Users find it difficult to explain their requirements in terms that developers can understand. Technical staff find it difficult to understand business requirements. According to Beynon-Davies et al. (1999), traditionally, the focus has been on rigorously engineering systems to satisfy a requirements document, whilst ignoring the fact that documented requirements may be inaccurate or incomplete.

The problems of waterfall approach as presented by Pressman (2000) are

1. Real projects rarely follow the sequential flow that the model proposes.
2. It is often difficult for the customer to state all requirements explicitly.
3. The customer must have patience. A working version of the program(s) will not be available until late in the project time-span. A major blunder, if undetected until the working programme is reviewed, can be disastrous.

According to Carter et al. (1997), Waterfall life cycle model developments can be very time consuming and hence time-scales tend to be lengthy. The major draw-back of this approach when applied to business-oriented software development according to Howard (2002) is delivery of systems that meet user requirements at the time they were specified but failed to meet user needs at the time of implementation months or years down the lane. According to Cepra and Verner (1996) some of the problems with waterfall approach identified by researchers are high development costs and user communication difficulties typical of long

and cumbersome processes. Even with all these drawbacks, "the classic life cycle remains the most widely used procedural model for software engineering" (Pressman, 2000).

7.2 Rapid Application Development (RAD) Approaches

However, these drawbacks in the waterfall or traditional approach led software developers to look for new approaches for software development which are normally identified as Rapid Application Development(RAD) approaches. There is no universal definition for Rapid Application Development. But many authors agree that the origin of RAD is the book titled "Rapid Application Development" by James Martin in 1991 (Agarwal et al.,2000; Tudhope et al.,2001; Howard A., 2000; Beynon-Davies et al.,1999). The following definitions explain how different authors view the RAD concept.

"RAD refers to a development lifecycle designed to much faster development and higher quality results than those achieved with the traditional life cycle" (Martin, 1991).

"RAD is an incremental software development process model that emphasises an extremely short development cycle. The RAD model is a high speed adaptation of the linear sequential model in which rapid development is achieved by using the component-based construction" (Pressman, 2000).

"The merger of various structured techniques (especially the Data-driven information engineering) with *prototyping* techniques and *joint application development* techniques to accelerate system development" (Whitten et al., 2001).

"RAD can be characterized in two ways: as a methodology prescribing certain phases in software development and as a class of tools that allow for speedy object development, graphical user interfaces, and reusable code for client/server applications" (Agarwal et al., 2000).

"RAD is not a tool; it is a methodology designed to take advantage of development tools to help IS professionals to develop small, medium, big and enormous projects in a fraction of time required by other methodologies" (Alvarez et al., 2000).

"An iterative and contingent approach to interactive software development that is characterised by large amounts of user involvement, the use of incremental prototyping and product-based project management" (Beynon-Davies and Holmes, 2002).

"Although RAD is marketed as a new approach, it is, in fact, a blend of existing techniques, albeit in a relatively new form" (Barrow and Mayhew, 2000).

From the above definitions, it is clear that software developers understand RAD in different ways. The following statement sums up the present level of understanding of RAD. "Some developers understand RAD to mean incrementally speeding up each stage of the traditional software life cycle. To others, RAD is primarily about automation and extensive use of software development productivity tools. For some, RAD will always

stand for Rough and Dirty development-an excuse for side tracking the disciplines of software engineering standards. Many believe RAD is suitable for small, relatively low-key projects. Others argue that RAD principles and techniques (Joint Application Development, time-boxing, prototyping and clean rooms to name a few) can be applied to any software project" (Howard, 2002). Even with all these ambiguities it is worthwhile to explore two specific methodologies popular in USA and UK.

7.2.1 Rapid Application Development as Proposed by James Martin

The key aspect of RAD is fast development. Martin (1991) argues that the definition for software quality, i.e. "confirming to the written specifications as effectively as possible", used under the traditional approach is wrong and the appropriate definition should be "meeting the true business (or user) requirements as effectively as possible at the system comes into operation." According to Martin (1991) four fundamental aspects of fast development exist: tools, methodology, people and management.

The tools available for IS developers for fast development are Fourth generation languages, Prototyping tools, CASE (Computer Aided Systems Engineering) tools and Code generators. "Some authorities advocate *prototyping*; some advocate *reusable design and code*; some advocate *end-user workshops for planning and design*; various authorities advocate *fourth generation languages, CASE tools, code generators, reverse engineering, a repository, automatic rule based validations and co-ordination, data modelling, process modelling, specialized implementation teams*, and so on. Any of these is valuable if used correctly. However, in synergistic combinations, they become more powerful and bring revolutionary change to the development process" (Martin, 1991).

There are four phases of RAD life cycle. They are:

Requirement Planning Phase;

User Design Phase;

Construction Phase;

Cutover Phase.

One of the most important differences between lifecycle designed for modern development tools and classical life cycle is that, today, the intended users of the system should be involved at every stage.

As identified by the editor of IEEE Software (2000), two of the best practices on Software Engineering in the 20th Century are incremental development and User Involvement. "We've seen tremendous developments in the past several years in techniques that bring users more into the software design process. Techniques such as JAD sessions, user interface prototyping, and use cases engage users with product concepts in ways that paper specifications simply cannot. Requirements problems are usually listed as the #1 cause of software project failure; these techniques go a long way toward eliminating requirements problems" (McConnell, 2000).

The people involved in the RAD process must be a special and highly trained team. For fast development excellent tools are needed, but people must know how to use the tools and work together as closely knit teams trained to use a well designed methodology.

To be as effective as possible, the management of an IS organization should identify *all* the good techniques and combine them. It should employ *all* the techniques that can improve a fast development lifecycle and try to make *all* of them work.

As presented by Beynon-Davies et al. (1999) common components of RAD approaches discussed in the literature are:

Joint Application Design (JAD) - RAD is done by small teams of 4-8 people consist with developers, users and managers. It uses JAD workshops at various points in the development process.

Rapidity of development - RAD projects seem to be typically relatively small scale and of short duration. It has been suggested that no more than six man-years of development effort should be devoted to any particular RAD project.

Clean Rooms - Provide places free from everyday work to developers and JAD workshops are conducted in places away from the business.

Time Boxing - If projects start to slip, the emphasis in RAD projects is on reducing the requirements to fit the time box, not in increasing the deadline.

Incremental Prototyping - The developers after some initial investigation, construct a working model that they demonstrate to a representative user group. The developers and the users then discuss the prototype agreeing on enhancements and amendments. This cycle is

usually repeated at least three times in RAD projects, until the user is satisfied with the system.

Rapid development tools - RAD demand good support from tools for rapid developmental change. This normally means some combination of fourth generation languages,

7.2.2 Dynamic System Development Method (DSDM)

Another popular RAD approach, especially in the UK, is Dynamic Systems Development Method (DSDM). It is a non-proprietary RAD method produced by the DSDM consortium. With the intention of becoming UK and international standard for RAD, DSDM Consortium was formed in January 1994 (Howard, 1997; Beynon-Davies et al., 1999).

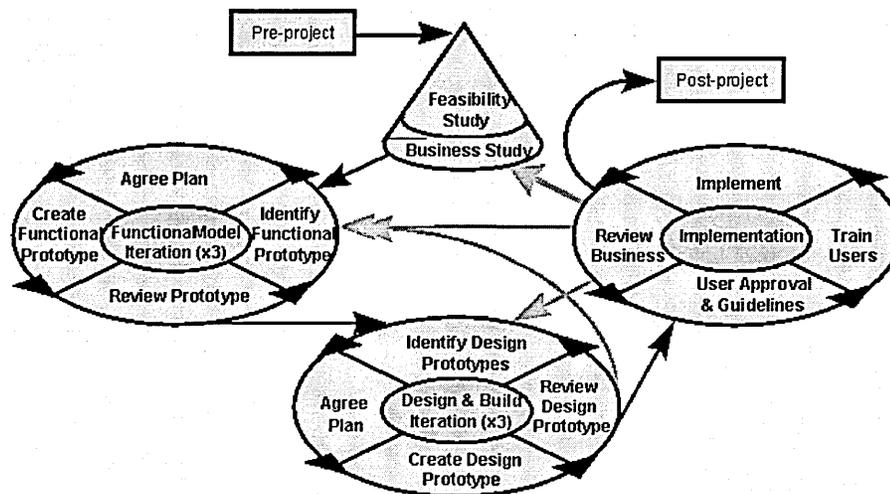
According to the DSDM Manual, a fundamental assumption of DSDM approach is that nothing is built perfectly first time but that 80% of the solution can be produced in 20% of the time that it would take to produce the total solution. DSDM assumes that all previous steps can be revisited as part of its iterative approach. Therefore the current step need to be completed only enough to move to the next step. DSDM is based on nine principles which are given below.

1. Active user involvement
2. DSDM teams must be empowered to make decisions.
3. The focus is on frequent delivery of products.
4. Fitness for business purpose is the essential criterion for acceptance of deliverables.
5. Iterative and incremental development is necessary to converge on an accurate business solution.

6. All changes during development are reversible.
7. Requirements are baselined at a high level - "Freezing" and agreeing the purpose and scope of the system at a level that allows for detailed investigation of what the requirements imply. More detailed baselines can be established later in the development, although the scope should not change significantly.
8. Testing is integrated throughout the life cycle.
9. A collaborative and co-operative approach between all stakeholders is essential.

There are five phases of development within a DSDM project (See Figure 7.2).

Development Process Framework



© DSDM Consortium 2002

Building Business Agility: Delivery Through Synergy with DSDM
www.dsdm.org

Figure 7.2: The DSDM life cycle (www.dsdm.org).

1. Feasibility study - This phase considers the feasibility of the project in business and technical terms as well as the suitability of the project for a RAD approach.
2. Business study - This phase defines the high level functionality and the major business entities affected.

3. Functional model iteration - This phase is used to construct and demonstrate the required functionality using a working prototype.
4. System design and build iteration. - This phase is used to refine the functional prototype, particularly to meet non-functional requirements.
5. Implementation - The implementation phase includes the handover to users followed by a review of the project's success.

Some of the common features of two RAD approaches explained above are extensive user involvement, iterative nature of the process, use of prototypes and emphasis on time over the functionality.

7.2.3 Issues in adapting RAD approach

The main emphasis of RAD is on shorter delivery time. It is the common belief that speed and quality do not move in the parallel direction. "RAD is accused of being anti-quality. It is commonly believed speed and quality are incompatible in software development. On the other hand, supporters of RAD argue that modern quality principles are inherent in RAD. Fitness for purpose, avoiding waste, getting things right the first time, individual responsibility for quality, and meeting customer requirements is as engraved in RAD as in quality (Howard,2002). According to Howard (1997), DSDM is not recommended for real-time applications or for computationally complex systems. One of the major concerns of RAD may be high cost of RAD when compared to conventional development due to maintaining clean rooms and greater degree to which business users are involved in RAD projects (Howard, 2002: Beynon-Davis et al., 1999).

	<i>Traditional Lifecycle</i>	<i>RAD lifecycle</i>
Objective	<ul style="list-style-type: none"> ➤ Developers try to develop a software which will satisfy agreed specifications. ➤ Primary focus is on delivering functionality 	<ul style="list-style-type: none"> ➤ A small team of users, developers and managers try to satisfy business requirements. ➤ Prime focus is on short delivery time
Stages	<ul style="list-style-type: none"> ➤ Design ➤ Code ➤ Test ➤ Delivery and support • Each stage is started after satisfying the preceding stage 	<ul style="list-style-type: none"> ➤ Requirement Planning phase ➤ User design phase ➤ Construction phase ➤ Cutover phase • Iterative process
Process	<ul style="list-style-type: none"> ➤ First specifications are written ➤ Then the detailed design is done ➤ Then the code is written ➤ Substantial time is taken to debug the code 	<p>with I-CASE tool</p> <ul style="list-style-type: none"> ➤ A detailed design is built on the screen of an I-CASE tool ➤ Code is generated from it and run immediately
People Involved	<ul style="list-style-type: none"> ➤ Team of analysts design the application ➤ A separate team of programmers may code and debug it 	<ul style="list-style-type: none"> ➤ IS professionals do the detailed design and code generation of one transaction after another, using the I_CASE¹ toolset. They may show each transaction, as it is built, to end users and make adjustments to it. ➤ End users are closely involved during the construction phase. They validate the screens and design of each transaction as it is built
Duration	<ul style="list-style-type: none"> ➤ Duration of the project depends on the functions to be programmed ➤ If project start to slip, emphasis on extending deadline. 	<ul style="list-style-type: none"> ➤ Functionality of the software is limited by the available time. ➤ If projects start to slip, emphasis is on reducing requirements to fit the time box.
User Involvement	<ul style="list-style-type: none"> ➤ User involvement is limited to the specification building and testing stages 	<ul style="list-style-type: none"> ➤ Users are involved in each stage
When a tangible product is displayed?	<ul style="list-style-type: none"> ➤ No tangible product until end of the development process 	<ul style="list-style-type: none"> ➤ Working prototypes are built
Possibility of changing requirements	<ul style="list-style-type: none"> ➤ Specifications are frozen 	<ul style="list-style-type: none"> ➤ Specifications can be changed

Table 7.1 : Differences between traditional and RAD software development approaches

¹ I-CASE tools are explained later in this chapter

Under RAD approach, changes to the system requirement will be frequent. According to Carter (1997), due to the lax way that requirements tend to be managed, the scope of project can easily wander as new requirements are thrown up during development or at the end of a time box. Table 7.1 summarises the differences between traditional approach and the RAD approach.

The fundamental difference between traditional development and DSDM is shown in figure 7.3.

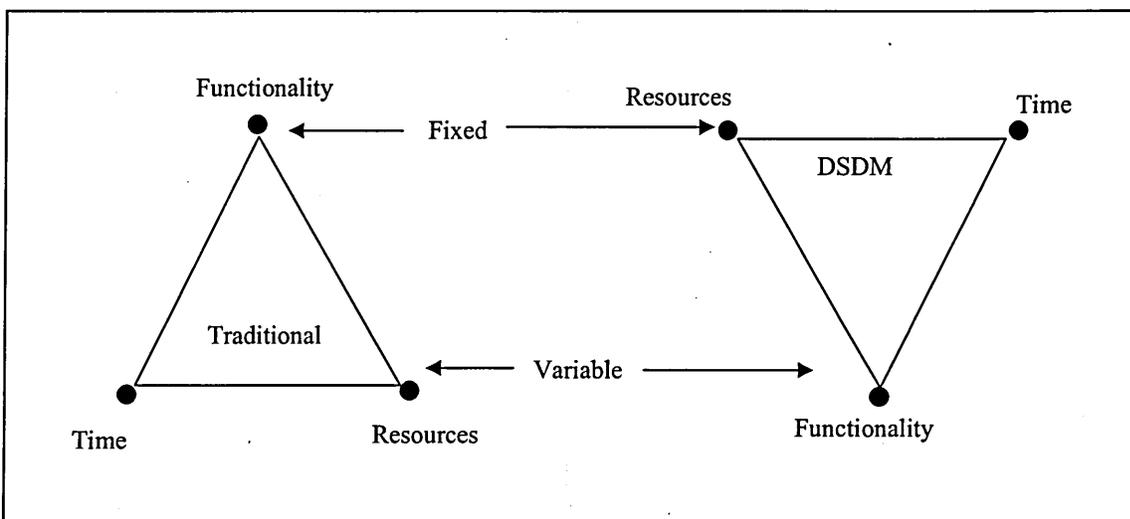


Figure 7.3: Comparison between traditional and DSDM development approaches.

www.dsdm.org

7.2.4 Tools under RAD Approaches

A number of different tools are available for developers who adopt RAD approach to accelerate the software development process. Two of such tools are explained below.

7.2.4.1 Prototyping

Prototyping is one of the most important tools cited in the RAD literature. But its applications are not limited to RAD. Prototyping could be used under the traditional approach as well.

As stated by Andriole (1991) the primary assumption that prototypes make is that interactive systems cannot be developed easily, quickly, or without input from prospective users. Prototyping is defined as “a technique for building quick and rough version of a desired system or parts of that system” (Martin, 1991). There are two major types of prototypes: *throwaway*, in which the prototype is discarded and not used in the delivered product, and *evolutionary*, in which all or part of the prototype is retained. (Gordon and Bieman, 1995; Martin, 1991; Andriole, 1991; Pressman, 2000). According to Cepura and Verner (1996) there are three types of prototypes.

- *Mock-up design prototyping*: produces throw-away demonstration prototypes. Mainly deal with input and output and usually end up as documentation.
- *Exploratory or working model prototyping*: produces requirements models for the final system. They help to visualize and experiment with functions or process with the prototypes usually ending up as part of the specification.
- *Evolutionary prototyping*: uses the prototype as the final system after a series of iterations based on user feedback.

According to Reilly (1995) evolutionary prototyping extends traditional prototyping to include prototyping during analysis. These prototypes evolve into functioning prototypes that eventually become the productive system.

As stated by Martin (1991), prototyping is particularly valuable in the following situations.

- There is a scope for user creativity to improve the system.
- Users are unsure of exactly what they want
- The system changes a basic business operation.
- An end-user dialog should be tried out with the users to see if it can be improved.
- The users do not understand all the impacts of the new system.
- The functions are subtle, and the users understand them better than the analysts.
- Screens and reports should be checked with management to see if they can be made more useful or easy to use.
- The users have difficulty expressing all the system requirements.
- The prototype may act as a catalyst to elicit alternative ideas.
- The relative merits of alternative solutions need to be explored.
- Experimentation may be done to achieve better business practices.

After analysing 39 cases of software development with prototyping Gordon and Bieman (1995) have identified following problems with prototyping.

- A prototype can demonstrate functionality that is not possible under real-time constraints, and this problem may not be discovered until long after the prototype phase is complete.
- Given too much access to prototype, end-users may equate the incompleteness and imperfections in a prototype with shoddy design. It give users the unrealistic expectation that there would be a complete and working system in a short time.
- A prototype developed quickly, massaged into the final product, and then hurriedly documented can be very difficult to maintain or enhance.
- Prototype development can be time consuming, specially when the purpose and scope of the prototype is not initially well defined.

Pressman (2000) has identified two problems of prototyping.

- The customer sees what appears to be a working version of the software
- The developer often makes implementation compromises in order to get a prototyping working quickly.

According to Martin (1991), prototype is fundamentally different from a paper description. It is real and manipulable. It can be adjusted and modified. The would-be users get a feel for what their system will be like. Its flaws are visible and tangible rather than buried in boring text. According to Verner and Cepra (1997) practitioners view prototyping as providing better communication with users, as providing more flexible designs, as better for

the early detection of problems, and as improving communication between IS personnel. Prototyping is one solution to the problem of unclear requirements. According to Martin (1991), Prototyping does so much to solve the problem of inadequate communication between designers and users.

Prototyping can be used under RAD as well as other software development approaches. But in the case of RAD it is a compulsory component. According to Martin (1991), prototyping in RAD lifecycle is fundamentally different from prototyping in some lifecycles in that the prototype must be part of the evolving system. Sometimes prototypes are built with a tool different from the final development tool and are eventually thrown away; in RAD lifecycles, they should be built *with* the final development tool so that they pass directly from the User Design Phase to the Construction Phase.

7.2.4.2 Computer Aided Software/System Engineering (CASE) Tools

Another widely discussed topic in the current software development literature is CASE tools. The main objective of the CASE is to automate the whole or part of the software development process. A simple definition for CASE is "CASE technologies are tools that provide automated assistance for software development" (Lending and Chervany, 1998). A more comprehensive definition would be "Computer-aided systems engineering (CASE) tools are software programs that automate or support the drawing and analysis of system models and provide for the translation of system models into application programs" (Whitten et al., 2001). As in the case of prototyping, CASE too may be used in RAD as well as in other approaches. As per Pressman (2000), CASE "can be as simple as a single tool that supports a specific software engineering activity or as complex as a complete

'environment' that encompasses tools, a data base, people, hardware, a network, operating systems, standards and myriad of other components". According to Huang (1998) CASE is the use of computer-based support in the software development process; a CASE tool is a computer -based product aimed at supporting one or more software engineering activities within a software development process. CASE tools can be used in different phases of the software development process. According to Post et al., (1998), various CASE tools now support several different aspects of systems development, including but not limited to: requirements elicitation and analysis, design and communication, enforcement of standards and methodologies, prototyping and RAD, reverse engineering, and software maintenance and re-engineering.

Some vendors have specialized in building code generation without front-end design and analysis tools. Others have built planning, analysis and design tools without a back-end code generator. However, what is needed is full integration between the CASE front-end tools and the generator so that code is automatically generated from the front-end tools. The term I-CASE/ICASE (Integrated CASE) is used to such a tool to differentiate it from normal CASE tools (Martin, 1991;Banker and Kauffman, 1991;McMurtrey et al., 2000;Subramanian G.H. and Zarnich, 1996). Oracle's *Designer 2000*, Platinum's *Erwin*, Rational's *ROSE*, Popkin's *System Architect 2001*, Sterling's *COOL* product family, Visible Systems' *Visible Analyst* and Visio's *Visio Enterprise* are some of the available commercial CASE tools (Whitten et al., 2001).

Subramanian and Zarnich (1996) provide a comprehensive list of features available in ICASE Tools. According to them, ICASE Tools provide support for

-
- Creating a model of user requirements in a graphical form;
 - Creating software design models for both the data and procedural code;
 - Error checking, consistency checking, analysis, and cross-referencing of all system information;
 - Building prototypes of systems and establishing a simulation of the system;
 - Enforcing organizational standards for specification, design, and implementation activities in the system development life cycle;
 - Generating code directly from the design models;
 - Providing automated support for testing and validation;
 - Providing support for reusable, software components-in the form of designs, code modules, and data elements;
 - Providing interfaces to external dictionaries and databases
 - Reengineering, restructuring, and reverse engineering of existing systems; and
 - Storing, managing, and reporting system-related information and project management information.

The most cited advantage of CASE tool is the improvement of software development productivity (Post et al., 1998; Martin, 1991; Banker and Kauffman, 1991; Lending and Chervany, 1998; Sommerville, 2001). Over the past years, many features had been added to CASE tools. According to Post et al. (1998), these additional features have increased the complexity of learning and using CASE tools. Another factor prohibiting the use of CASE is its cost (Banker and Kauffman, 1991).

7.3 Summary

Software is a computer programs and its associated documentation. In that sense a simulation model is also a piece of software. There are few approaches to software development. The most widely used approach is called the Waterfall or Linear Sequential Approach. Under this approach, software is developed in stages, which are basically design, code, test and deliver/support. It is required to complete one stage before the proceeding stage is started. This approach is not capable of handling subsequent changes of requirements. The problems with this approach led to the development of 'Rapid Application Development (RAD)' approach. The key aspect of RAD is fast development. Even though there is no universal definition for RAD, most of the RAD approaches found in the literature include components such as Joint Application Development(JAD), Prototyping, Time Boxing, CASE tools and Clean Room. The most important differences between RAD and traditional approach are the high user involvement and focus on the duration of the project development over the functionality in the latter case.

The software development process in general and the different techniques in accelerating the software development process were discussed in this chapter. As the objective of the present research is also to accelerate the simulation model development process, RAD has the potential for being used in achieving that objective. A new framework to accelerate the simulation model development process by synthesising the knowledge gathered through various research strategies such as case studies and questionnaire survey and software engineering principles are presented in the next chapter.

Chapter Eight

Development of the New Methodology

In chapter six it was found that there is a need for a new methodology for accelerating the simulation model development process. Then the next step is to propose the new framework. Software engineering methodologies, which were explained in chapter seven, were also used in developing the new framework in this chapter.

8.1 Direction for a New Framework for Accelerating Simulation Model Development

The objective of this section is to find a direction, i.e. to build a foundation, for a new methodology to accelerate the simulation model development process by putting together all the knowledge gathered through case study, questionnaire survey and literature review.

As initially suspected through reading the literature and confirmed through the questionnaire survey model development occupies the greatest proportion of time spent on a simulation project. Even though there is a nine year time difference between the graphs shown in figures 6.1 and 6.2, it is important to note that model development process continues to consume almost half of the time needed to complete a simulation project. Therefore not only current literature but also empirical evidence justifies the effort placed in researching rapid simulation model development.

The participants of the questionnaire survey agree with the statement that new issues arose during the experimentation stage led them to re-build new models/modify built models. The

most probable reason for this phenomenon may be that the model built by the modeller may not be the model required and explained by the user. Therefore, the fact that the lack of understanding of the system between the user and the modeller contributing to delay the simulation model development process, which was found under case study research and literature review, is also supported by the findings of the questionnaire survey as well. The other factor identified through literature review and the case study is the long duration of time needed to programme and test the simulation model in the selected simulator or the simulation language. Another factor which was to be considered in developing the new methodology was that the contemporary research on rapid simulation model development concentrated mainly on improving the efficiency of the process. But it was found that not improving the efficiency, i.e. developing the model at a shorter time, but also the effectiveness, development of the model required by the user, of the process is also important in accelerating the model development process. Therefore, it is apparent that if it is possible to improve the understanding between the user and the modeller and to reduce the programming burden, then the time required to develop the simulation model will be reduced.

The best way to reduce the understanding gap between the user and the modeller is to develop a tool for the benefit of the user to develop the model by himself. However, the findings of the questionnaire survey revealed that still a modeller with a simulation background plays a major role in the simulation model development process. Further, it was found that even experienced simulationists need the help of general programming languages and packages in developing the computer simulation model. In this regard, both simulationists and the user must be a part of the methodology. However, it is important to drastically increase the presence of the user in the process.

In this context there must be three major components of the proposed methodology. Firstly, a component to improve the participation of the user in the model development process. Secondly, a component to improve the understanding of the system between the user and the modeller and thirdly, a component to reduce the programming burden of the modeller.

Another important finding of the questionnaire survey which was kept in mind was that respondents agreed with the statement that they develop a conceptual model before building a simulation model. Further, findings revealed that conceptual model building is the most difficult task of the simulation project. Therefore it seemed that if a computer aided tool can be provided for building the conceptual model, then the model expected by the user can be built in shorter period of time.

As mentioned earlier, since a simulation model is also a piece of software, how the techniques used in accelerating the software development process in general can be used in developing the new methodology are discussed in the next section.

8.2 Use of Software Development Techniques in Simulation Model Development

The following analysis of the simulation project life cycle shows that it is closer to the waterfall life cycle approach of software development than the other approaches discussed in chapter seven. Tye (1999) also expressed same opinion. Table 8.1 shows a comparison between the waterfall approach of software development and simulation project life cycle. Arrows show the analogues stages in two approaches.

Attribute	Simulation project life cycle	Waterfall approach
Objective	Build a model according to the identified requirements	Construct a program to satisfy the identified requirements
Stages	Identification of the problem Model Building Experimentation Implementation	Design Code Test Delivery/support
Final Output	User will see the working model only after the complete model is built	The working model of the program is delivered to the user at the end of the project time.

Table 8.1: A comparison between waterfall approach and simulation project life cycle.

Simulationists who have involved in all three case studies which were explained in chapter five have used the waterfall approach. However, there are several factors which may cast a doubt on the effectiveness of using the waterfall approach on the simulation model development. Firstly, the success of the waterfall approach depends on the clear expression of the requirements by the user and understanding of them by the programmer. But, in the case of a simulation project, even though the user may have some idea or a vision about the system, he may still not be able to precisely express the system to the modeller even with the help of diagrams, especially in the case of to-be systems. On the other hand, the modeller's lack of domain knowledge may lead to a misunderstanding of the requirements expressed by the user. Secondly, one of the major problems with the simulation project

cycle is that the user would see the final model only after the complete model is built. Therefore, the user does not have an opportunity to check whether a model representing the desired system is being developed through the model development process. This may have been one of the reasons for identifying restricted flexibility as one the major limitations of simulation software by the users in user surveys in 1992 and 1997 (Hlupic, 1999). Thirdly, simulation projects usually have a tight time schedule within which the project must be finished. But waterfall approach follows distinctive stages and normally it may take a long time to complete the project. The development of the **right** simulation model within a shorter period will provide a substantial time out of the project time for the user to spend on experimentation and thereby come up with better results.

In this background, it was more appropriate to use Rapid Application Development (RAD) approach (explained in detail in chapter seven), which is characterised by emphasis on fast development, opportunity for changing requirements during any stage of the life cycle, active user involvement, testing throughout the development process and development of a product to meet the business need rather than meet the requirements, than the waterfall approach in developing new methodology consisting with the components mentioned in section 8.1. This developed new methodology is explained in the next section.

8.3 A New Methodology for Accelerating Simulation Model Development Process

The basic thrust behind the proposed methodology is the improvement in understanding of the system to be simulated between the user and the modeller, thereby developing the right simulation model, i.e. a model represents the actual system, first time. Another major factor

kept in mind when developing the new methodology was that the user, the domain expert, is not necessarily familiar with the simulation terminology and the modeller, simulation expert, is not familiar with the system to be simulated. In addition to improve the understanding, it was tried to automate the programming activities as much as possible to reduce the programming burden of the modeller. In section 8.1 it was identified that there must be three components in the new methodology.

Components to be included into the new methodology	Concept selected from RAD approach	<i>Purpose</i>
Component to improve the participation of the user in the model development process	 Joint Application Development Team (JAD) 	<i>Allow the user and the modeller to develop the model together</i>
Component to improve the understanding of the system between the user and the modeller	 A CASE tool 	<i>Develop a prototype of the system to be simulated which will serve as the conceptual model of the system</i>
<i>component to reduce the programming burden of the modeller.</i>	 Programme generator 	<i>Translate the conceptual mode into the computer simulation model</i>

Figure 8.1: Use of RAD concepts in the new methodology

Figure 8.1 shows how the different concepts selected from RAD approach of software development matched with the components to be included into the new framework. The rationale of including those components into the new methodology and the purpose of each component are explained in next sections.

8.3.1 Joint Application Development Team

It was mentioned earlier that it is probable that there will be a gap in understanding between what the user expected and what the modeller developed model due to lack of understanding of requirements. "In order to remove this gap in understanding [between the user and the modeller], it may be preferred that the clients build their own models" (Robinson, 1994). But even with the advances in simulators, simulation users have identified difficult to learn as one of the major limitations of simulation software at the simulation user's survey (Hlupic, 1997). Results of the questionnaire survey also revealed that even experienced simulationists use non-simulation languages such as C++ and VB and packages such as MS_Excel and Access in developing simulation models. Therefore it is not cost effective to train occasional users in using simulation software to develop models by themselves. In general, firms do not carry out simulation projects on regular basis. Once a facility is designed, it may not be changed for some time. In that case users may prefer not to spend their valuable time learning a new software which could not be used for some purpose other than simulation, a job that they do not usually perform in the work place under their normal duties. Therefore, one of the premises of the proposed approach is that the modeller cannot be removed from the scene and he has to play a major role. But still a high degree of user involvement in the model development process is crucial to a successful outcome. Therefore, it is proposed to form a Joint Application Development (JAD) consist with the user and the modeller to carry out the simulation model development process under the new methodology. This team would conduct Joint Application Development (JAD) sessions/workshops throughout the model development process. A new tool would be developed to help the members of the JAD team to communicate effectively among themselves during those JAD sessions. This would

eventually lead to improving the efficiency as well as the effectiveness of the model building process. This tool is explained in the next section.

8.3.2 A CASE Tool to Develop Prototypes

As pointed out by Fairly and Thayer (1997), the traditional approach of software development does not facilitate communication among users, buyer¹, and developer; nor does it emphasize the importance of specifying the operational requirements for the envisioned system. The goal of software engineering is to develop and modify systems that satisfy user needs, on schedule and within budget. Accurate communication of operational requirements from those who need a software-intensive system to those who will build the system is thus the most important step in the system development process.

Detecting and correcting a mismatch which has arisen due to a misunderstanding of the system at the end of the simulation model development process is very costly. One way of narrowing the gap between the user expected model and the developer built model is developing a conceptual model at the early stage of the project. But, if it is to be successful, such a conceptual model should be developed in a common language which both the user and the modeller can understand. The best common language which can be used for developing the conceptual model is symbolic language which uses simple symbols such as boxes and arrows. However, the level of abstraction of the system needed depends on the requirements of the user of the conceptual model. For example a top manager of the firm

¹ The buyer is a representative of the user community (or communities) who provides the interface between users and developers.

may need only a general view of the system where as a manager at the floor level may need all the details.

According to Robinson (1994), a day on paper saves a month on a computer. He emphasizes the need for proper model structuring away from the computer. But, it is a cumbersome process which many modellers prefer to avoid. A paper based conceptual model will reduce the gap between the required model and the developed model up to a certain extent. But still, the user may not get as thorough understating as of the model he would have had the model been developed using a visual interactive model.

However, the use of prototyping approach and CASE Tools concepts used in the RAD approach in the simulation project will allow the JAD team to develop a conceptual model gradually on the computer screen.

According to Martin (1991) prototyping differ from paper descriptions on the following aspects.

- A prototype is real and manipulatable
- It can be adjusted and modified
- The would -be users get a feel for what their system would be like
- Its flaws are visible and tangible rather than burying in boring text.
- If right tools are used, prototypes can be created much more quickly than written specifications.

Therefore, it is proposed to use the prototyping tool of RAD approach in simulation model development in order to develop the conceptual model of the system on the computer screen. The first column of table 8.2 shows the situations where the prototyping is particularly useful (Martin, 1991), and the second column indicates the relevancy of the given situations to a simulation project.

<i>Situations where Prototyping is particularly useful</i>	<i>Relevancy to Simulation</i>
There is scope for user creativity to improve the system	✓
Users are unsure of exactly what they want	✓
The system changes a basic business operation	
An end-user dialog should be tried out with the users to see if it can be improved	✓
The users do not understand all the impacts of the new system	✓
The functions are subtle, and the users understand them better than the analyst	✓
Screens and reports should be checked with management to see if they can be made more useful or easy to use	✓
The users have difficulty expressing all the system requirements	✓
The prototype may act as a catalyst to elicit alternative ideas	✓
The relative merits of alternative solutions need to be explored	✓
Experimentation may be done to achieve better business practice	✓

Table 8.2: Relevancy of prototyping to the simulation.

In the case of simulation, the logical prototype of the system will serve as the conceptual model of the system to be simulated. In order to develop this conceptual model on the computer screen, a CASE tool named CASCOMoD², Computer Aided Simulation Conceptual Model Developer, is proposed under the new methodology. With the aid of the CASE tool, the JAD team can 'walk through' the conceptual model and make necessary amendments to it before the computer simulation model is built. The main objective of this

² Technical details of CASCoMoD are discussed in detail in Chapter Nine.

CASE tool is to provide a common platform for the user and the modeller to mutually exchange their understanding of the system to be simulated. This tool provides an opportunity for the user to express and verify the model without requiring to have any technical knowledge beyond what is required to perform their daily job functions. The conceptual model is developed by using the terminology of the users' application domain. On the other hand, modeller can express his understanding of the system to the user. Therefore this tool fulfils the following two roles.

1. To communicate the users requirements to the model developer.
2. To communicate a developer's understanding to users.

This will provide an opportunity for the user to see a visual model while it is being developed rather than waiting until the end of the model building process.

During the case studies, it was found that even though the modeller kept contact with one person during model development another person validated the developed model. Or else, due to some reason if a new person was contacted from the client organization, he came up with some modifications to the model which were not mentioned by the person who made regular contact. It is not necessarily a waste of time. Obtaining different views and perceptions regarding the system held by relevant stakeholders of the system will help to develop a better model. But, if these differences of opinions are surfaced at a latter part of a project they will make the modeller's task difficult and hence delay the project. Therefore, this tool will have a facility to make a presentation of the developed conceptual model to all the stakeholders to obtain their consensus before the computer model is built.

The major difference between the model developed under the proposed approach and the model developed under the current approach will be that former is more user-oriented where as the latter is more simulation-oriented and developer-oriented.

8.3.3 Translation of the Conceptual Model into the Computer Model

A conceptual model will improve the understanding of the system. However to get the full advantage of the RAD approach, the developed prototype should be an 'evolutionary' prototype rather than a 'throw away' prototype. Otherwise the time spent on developing the conceptual model may be squandered. In the case of a paper based conceptual model, the modeller has to develop the computer model from scratch, even though s/he has spent a great deal of time developing the conceptual model on a piece of paper. Besides, there is no guarantee that the developed computer model reflects the conceptual model because these two are independent of each other. Further, there is a need for reducing programming burden of the modeller as well. This can be achieved by automating the model development activities as much as possible. Therefore, the third component of the proposed methodology is a tool for translating the developed conceptual model into a simulation language specific computer model.

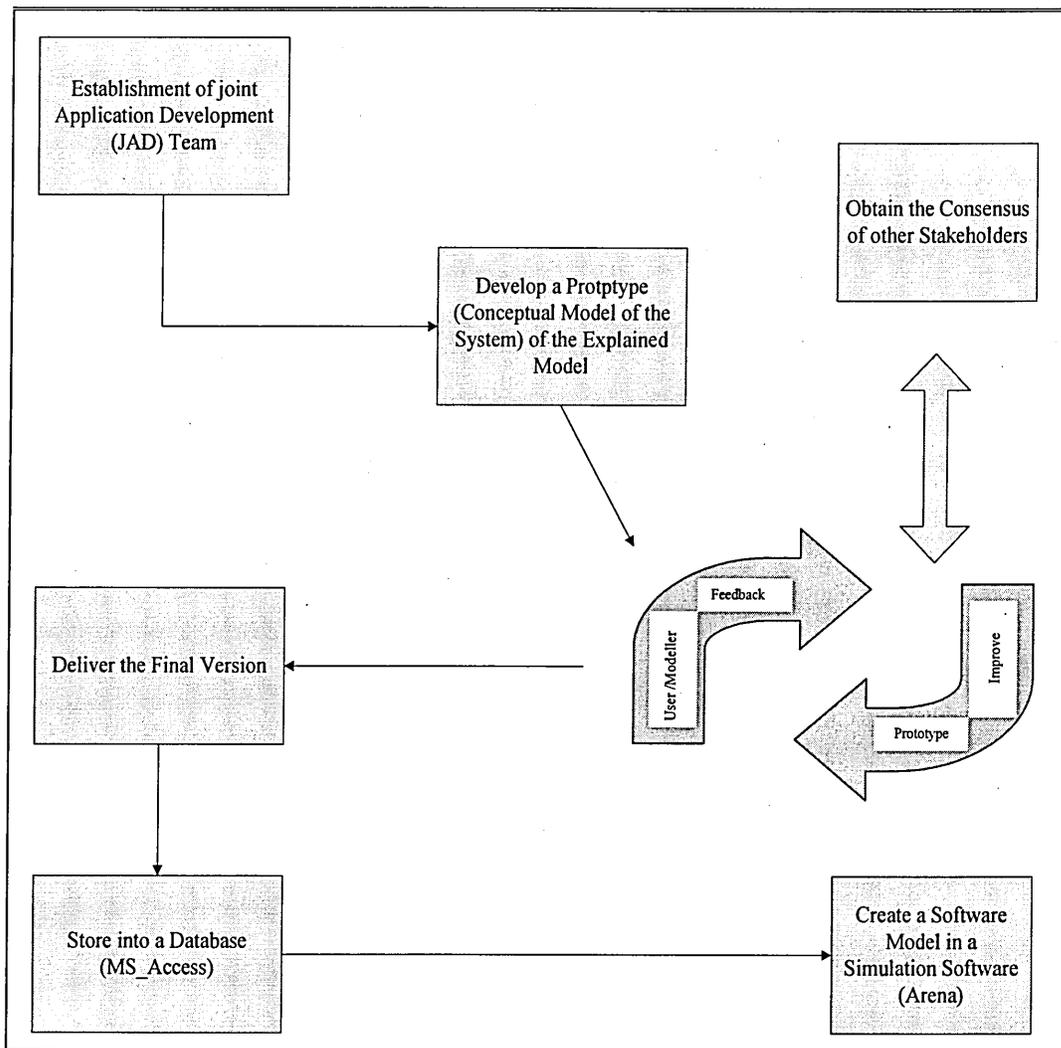


Figure 8.2: Schema of the proposed method

Once the conceptual model is finalized, i.e. after obtaining the consensus of all the stakeholders, the conceptual model is saved into a database in a format which can be identified by the simulation software. From the questionnaire survey, it was found that Arena is the most popular simulation software used by simulationists. (40% of the respondents use Arena). Therefore, an interface program for translating the conceptual model to the simulation software was developed for Arena. Figure 8.2 shows the schema of the proposed methodology.

Figure 8.3 summarises the rationale behind the new methodology.

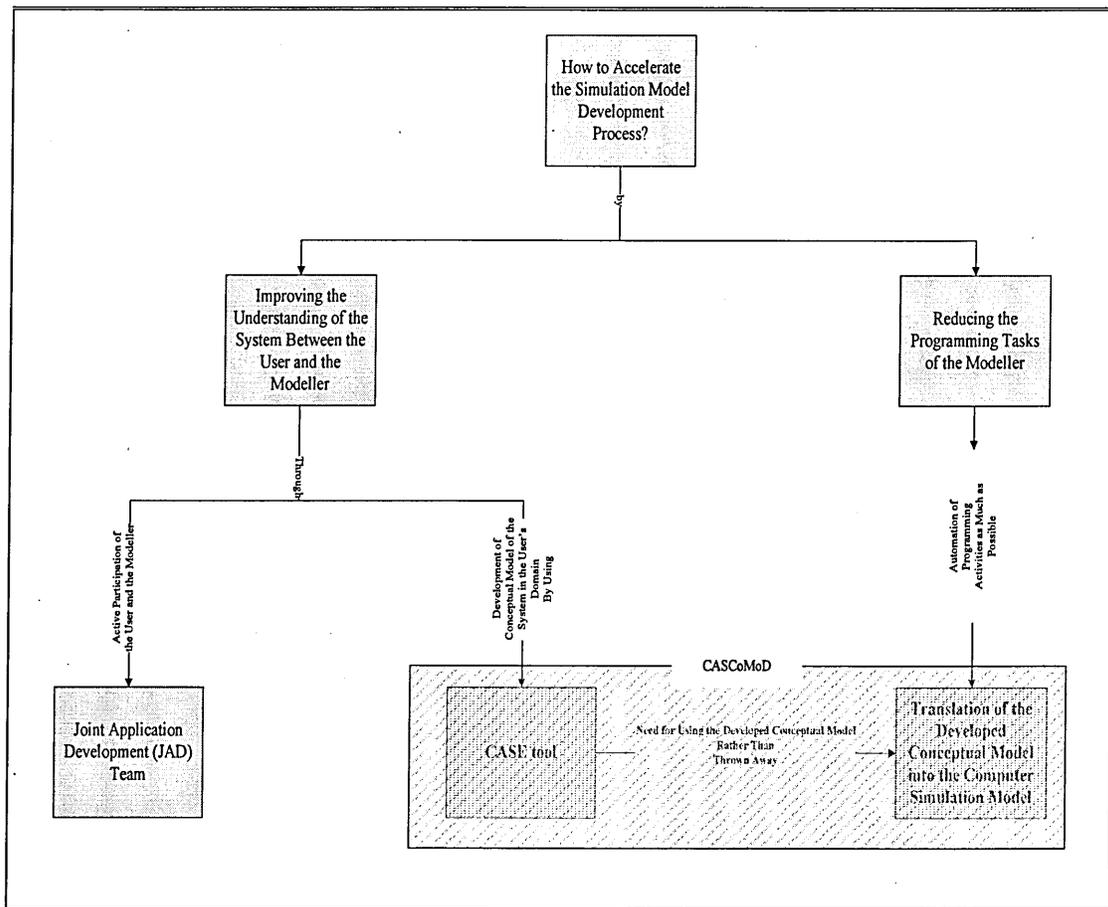


Figure 8.3: Rationale behind the new methodology

How the proposed approach accelerates the simulation model development process is explained in the next section.

8.4 How the Proposed Methodology Accelerates Simulation Model Development Process.

This section explains how the proposed approach will accelerate the simulation model development process when compared to the methods currently used by simulationists. Earlier in this chapter it was mentioned that the proposed methodology accelerates the simulation model development by increasing the effectiveness as well as the efficiency of

the simulation model development process. It was identified that in the context of simulation, effectiveness is building the right model to represent the actual system and efficiency is building the model in a shortest period of time. Building the model required by the user at the first attempt will substantially reduce the time wasted due to rework of the model. It was identified that one of the reasons for lengthiness of the model development process is the semantic gap between the model developed by the modeller and that required by the user which leads to the re-building of the model. Adaptation of the proposed methodology will make sure that the right model is built first time by improving the understanding between the user and the modeller. If the proposed method is adopted, the logic of the model is agreed by the user, the modeller and other stakeholders during the conceptual model development phase. This helps to develop a computer simulation model which is more likely to represent the system required by the user. This understanding of the system will lead to the elimination or reduction of rework. Therefore, number of iterations needed to obtain the final model from the first developed model will be reduced and thereby the time required to develop the simulation model is also reduced.

Secondly, since the user and the modeller are engaged in the model development process actively and effectively under the proposed method, any problem can be clarified or conflict arisen during the model development process can get solved instantaneously. In contrast, the usual practice is for the modeller and the user to work in two different places and maintaining contact through occasional meetings. So the modeller will have to stop developing the model and wait until the clarification is obtained.

Thirdly, the proposed methodology improves not only the effectiveness but also the efficiency of the simulation model development as well. This is achieved by reducing the

actual programming task. According to Martin (1991), prototyping in a RAD lifecycle is fundamentally different from prototyping in some lifecycles that the prototype must be a part of the evolving system. The conceptual model developed under the present methodology is rather an evolutionary type conceptual model than a throw away type conceptual model, i.e. the conceptual model is translated into the computer simulation model. Once the conceptual model is translated into the computer model, the core of the computer model is completed. Since the modeller is involved in the conceptual model development process from the beginning, he is quite familiar with the model. The modeller's task is limited to making enhancements as necessary to the automatically generated computer model. A conceptual model may be developed under the present practice as well. But the computer model is to be developed from the scratch again. Therefore, there is no guarantee that the developed computer model coincides with the conceptual model under the present system as they are independent of each other. But the automatic translation under the new approach will ensure that the computer model reflects the conceptual model.

Fourthly, since the most of the programming code is automatically written by the transformation programme, the amount of time needed for verification is less than would traditionally be required. Time wasted due to errors in programming such as misspelling is also eliminated. Further, most of the cases, names of modules such as entities and resources are provided by the model transformation programme. This avoids the duplication of module names.

Fifthly, not only the time for verification but also time needed for validation of the model is reduced. Since the conceptual model has been validated by all relevant stakeholders, there

will be less validation of the computer simulation model required. It is closer to the model required by the user. Therefore, there should be hardly any need for modifications to the computer model.

Finally, as a separate page for each part type is created in both drawing the structural model and in developing the quantitative spreadsheet model, the proposed tool will serve as a checklist for user and the modeller to ensure that all the required information are gathered and included into the conceptual model. This will reduce the time wasted at the verification and validation stages due to omission of information at an earlier stage. Therefore, as a whole the proposed methodology reduces the time needed to build the simulation model. Figure 8.4 presents a comparison of the total time spent on model development under the present approach and the proposed approach.

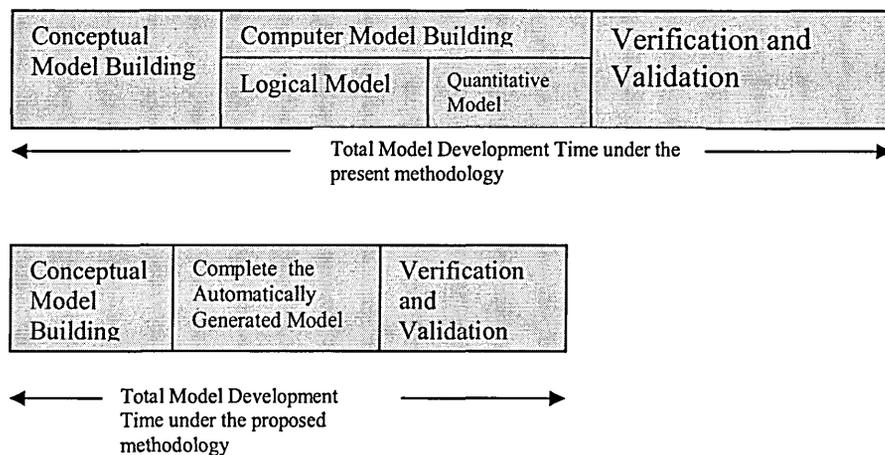


Figure 8.4: A comparison of simulation model development time between the proposed approach and the present approach.

8.5 Validation of the Proposed Methodology

As explained earlier, the new methodology was developed based on software engineering principles. Simulation, specifically discrete event simulation, has its roots in computer science. Therefore it is worthwhile exploring how the claims on findings of the research are validate in the field of computer science field.

8.5.1 Validation in Computer Science in General

There is no agreement among scholars (Denning, 2005; Khazanchi and Mukvold, 2000; Brooks, 1996; Tichy, 1998; Hartmanis, 1995; Xia, 1998) regarding whether the computer science defined as “the systematic study of algorithmic process that describe and transform information: their theory, analysis, design, efficiency, implementation, and application. The fundamental question underlying all of computing is, “What can be (efficiently) automated”” (Denning et al, 1988) meets the criteria of science defined as

- the state of knowing : knowledge as distinguished from ignorance or misunderstanding
- knowledge or a system of knowledge covering general truths or the operation of general laws especially as obtained and tested through scientific method. (Merriam Webster Online,2004)

Hence there is no consensus among scholars regarding the perspective on which the subject of validation is to be discussed in the discipline of computer science.

According to Xia (1998), to build a scientific theory it is needed first to formulate a hypothesis which is the understanding of the phenomena under investigation and then verify the hypothesis through tests. However, according to Hartmanis (1995), computer science advances are often demonstrated and documented by a dramatic demonstration rather than a dramatic experiment as in physical sciences. It is the role of the demonstration to show the possibility or feasibility to do what was thought to be impossible or not feasible. According to Tichy (1998) demos can provide proof of concepts (in the engineering sense) or incentives to study a question further. Too often, however, these demos merely illustrate a potential.

According to Tichy (1998), there are a number of well known computer science theories that have not been tested. For example, functional programming, object-oriented programming, and formal methods. Further, a study conducted by selecting a random sample of all papers published on ACM in 1993, he found that 40 per cent of the papers with claims that needed empirical support had none at all. In software related journals, this fraction was 50 percent where as in the non computer-science journals this figure was merely 15 per cent.

In an analysis of papers published in 1985, 1990 and 1995 in three computer science journals ICSE, Software and TSE, Zelkowitz and Wallace (1998) have identified 12 software engineering validating models(See table 8.3). In this study, they found that 27% of the papers had no experiment and assertion has become the most used method with 31% and case study method came second with 9.4%.

Validation Method	Category	Description	Weakness	Strength
Project Monitoring	Observational	Collect development data	No specific goals	Provide baseline for future; inexpensive
Case Study	Observational	Monitor project in depth	Poor control of later replication	Can constrain one factor at low cost
Assertion	Observational	Use ad hoc validation systems	Insufficient validation	Serves as a basis for future experiments
Field Study	Observational	Monitor multiple projects	Treatments differ across projects	Inexpensive form of replication
Literature Search	Historical	Examine previously published studies	Selection bias; treatments differ	Large available database; inexpensive
Legacy	Historical	Examine data from completed projects	Cannot constrain factors; data limited	Combine multiple studies; inexpensive
Lessons Learned	Historical	Examine qualitative data from completed projects	No quantitative data; cannot constrain factors	Determine trends; inexpensive
Static Analysis	Historical	Examine structure of developed product	Not related to development method	Can be automated; Applies to tools
Replicated	Controlled	Develop multiple versions of product	Very expensive; Hawthorne effect	Can control factors for all treatments
Synthetic	Controlled	Replicate one factor in laboratory setting	Scaling up; interactions among multiple factors	Can control individual factors; moderate cost
Dynamic analysis	Controlled	Execute developed product from performance	Not related to development method	Can be automated; applies to tools
Simulation	Controlled	Execute product with artificial data	Data may not represent reality ; not related to development method	Can be automated; Applies to tools ; Evaluation in safe environment

Table 8.3: Experimental Models for Validating Technology, Zelkowitz and Dolores (1998)

From the facts presented so far it is clear that validation in the computer science field cannot be considered in the same context as of exact science such as physical sciences. Basili (1996) identifies the reasons preventing the use of experimentation in the field of software engineering as follows. “Most of the technologies of the discipline [software engineering] are human based. It does not matter how high we raise the level of discourse of the virtual machine, the development of solutions is still based upon individual creativity, and so differences in human ability will always create variations on the studies.

This complicates the experimental aspect of the discipline. Unlike physics, the same experiment can provide different results depending on the people involved. This is a problem found in the behavioural sciences. Besides the human factor, there are a large number of variables that affect the outcome of an experiment. All software is not the same; process is a variable, goals are variable; context is variable. That is, one set of processes might be more effective for achieving certain goals in a particular context than another set of processes.”

So validation of the proposed methodology is disused in the light of the facts discussed above in the next section.

8.5.2 Validation of the Proposed Methodology

The objective of the proposed methodology was to accelerate the simulation model development process. However, it is impossible to experimentally validate, i.e. to conduct a hypothesis test, to establish that the proposed methodology requires less time for simulation model development when compared to the traditional development method. If it is to be tested statistically, the proposed method and the present method would need to be applied to the same project and with the time required for each method measured and compared. Not only the same project, but also the same set of people should be involved in the both situations. (Obviously this experiment needs to be performed several times for different projects in order to draw a statistically valid conclusion.) The proposed method anticipates reduction in time required by improving the understanding of the system. Once the model is developed under one methodology the modeller and the user gain an understanding of the system. Since they start the second phase of the experiment with that understanding obviously the time required for model development will be reduced. Further, as mentioned

in the previous section, even the same set of people, users and the modellers, may not behave in the same manner in two different situations. There is no guarantee that the same model is produced even by the same modeller with the same methodology in two different occasions. In other words it is impossible to have a control project for this experiment. Therefore, it is apparent that it is impossible to compare the time required to develop the simulation model under the proposed approach and the traditional approach, i.e. to set a hypothesis and to test it, and come to a statistical conclusion.

In this context, it was required to find alternative methods of validating the proposed method. Therefore, in this perspective the definition used for validating the methodology was “assessing the effectiveness of proposed development tools and methods in various environments” (Shull et al., 2001). Strategies used for validation are case study, literature review and expert opinion. From the case studies, literature review and questionnaire survey it was found that the two major reasons for delaying the simulation model development process are lack of understanding between the user and the modeller and the difficulty in programming. Obviously, if the proposed methodology can improve the understanding between the user and the modeller and it can ease the programming task of the modeller, the proposed methodology should accelerate the simulation model development process. Therefore, measurement of the above mentioned two factors were attempted during the validating process. Application of the above mentioned techniques to validate the proposed methodology are explained in detail in the following sections.

8.5.2.1 Case Study

The first strategy used was case study research. The developed tool was applied to the three cases explained in Chapter five. It was possible to successfully develop the conceptual

model using the developed tool and subsequently to translate the developed model into the computer simulation model. That proved the feasibility of the proposed tool. During the discussions with the model developers involved in each of the cases they agreed that the proposed tool will improve the understanding of the system between the modeller and the user. Further, they agreed that the automatic translation may reduce the programming burden by providing the appropriate modules. Since the names of the modules were provided by the tool, programming errors such as misspelling and duplication were prevented. However, they were cautious about restricting programming flexibility arisen due to the automatic translation of the conceptual model into the computer model. There may be more than one technique to accomplish the same task in developing the computer simulation model. For example, CASCoMoD used the 'DECIDE' module frequently in developing the route of the each part type where as the modeller may prefer 'SEQUENCE' module for the same task. However, once the conceptual model is automatically translated into the computer model, the modeller has to work with the model provided by the tool. This can be rectified by improving the translation programme of the tool to provide an option for the modeller to select modules to be included into the translated simulation model.

An earlier version which did not have a facility for including the quantitative model as part of the conceptual model was used in the case studies. At that stage it was thought that the most difficult task in the model development was building the logical model. However, the application of the new tool to case studies found that not only building the logical model but also amalgamation of quantitative model was also a time consuming task. Especially, in the case of push and pull comparison case, amalgamating of the quantitative data into the developed computer model was very time consuming as there were data

regarding 35 different part types and more than seven machines. Therefore, in the second version of the tool, which is explained in detail in the next chapter, there is a facility to include the quantitative data into the conceptual model by using a simple spreadsheet and subsequently converting that also into the computer model.

8.5.2.2 Literature Review

The next strategy used for validation was use of available literature. Since the advantages of RAD approach, CASE tool and prototyping were discussed in detail in chapter seven, they are not repeated here. However, a number of authors have cited the importance of visualization and graphics on understanding the complex situations. As noted by Kamada and Kawai (1991), pictorial representation of information has played an important role in human communication and recognition since time immemorial. Humans can grasp the content of a picture much faster than they can scan and understand a text sentence because they have the ability to recognise the spatial configuration of elements quickly. According to Tversky (2002) graphics provide additional way of representing information; two codes, pictorial and verbal are better than one. Graphics may save words by showing things that would otherwise need many words to describe. This especially holds for faces, maps, systems, and that are naturally spatial and hard to describe or visualize. Further, another function of graphic displays is to use space to organize information and to facilitate memory and inference. Graphics externalize internal memory. This has at least two benefits. The benefit to the individual mind is reducing the burden on memory and processing by off-loading. The benefit of a group of minds is joint consideration of the same set of ideas as well as collective revisioning of them. Finally, graphics have been used to promote inference and discovery by making the underlying structures and processes

transparent. However, simple graphics with less detail are often more effective than more realistic ones provided that they abstract the essential conceptual information.

According to Crapo et al. (2000), when text, i.e. symbols representing natural language, are placed on a page, which is inherently 2-dimensional and can be traversed by the eyes in either direction, it is rare for a person to be able to comprehend the meaning in other than a sequential fashion; line by line in the case of English. Diagrammatic representations may be distinguished from sentential representations by observing that information in a diagrammatic representation is indexed spatially, by location in two or three dimensions, and is processed by the visual portions of the brain. Further, according to them, people tend to create mental image to analyse information given in the text form. However, as the complexity of the image increases, peoples' ability to examine or manipulate the image to solve problems becomes increasingly inferior to peoples' ability to use an external visualization aid to solve the same problem. Further, creative insight and problem solving performance can be improved with appropriate visualization. Reasons they identify are, firstly, that visualization extends working memory by using the massively parallel architecture of the visual system to make an external representation function as an effective part of the working memory and secondly, by reducing the degree of freedom of expression, interpretation becomes easier, thus making diagrams more effective.

According to Aggrwal and Rezaee (1996), "TQM [Total Quality Management] advocates that the best approach to minimize defects is to understand user needs. Prototyping and diagramming tools provide efficient communication links between users and developers. Prototyping allows the users to evaluate the design and ask for modifications in early phases. Such an approach drastically reduces errors at an early stage. Users' continual

involvement in all phases of systems development is the best method to minimize errors and maximize user satisfaction.” They have explained how to achieve TQM in a general system development situation. The proposed methodology tries to do exactly the same thing in the simulation model development.

From the above literature, it is clear that well designed graphics provide a better understanding of the subject under consideration when compared to the textual description by enhancing people’s mental capacity. Not only that, by limiting the degree of freedom for interpretation, graphics allow a group of people to come to a close conclusion regarding the subject. Therefore, the idea of improving the understanding of a system by allowing the user to develop graphical conceptual model is supported by the available literature.

Another argument put forward under the proposed methodology is that the Joint Application Development approach accelerates the simulation model development process. According to Carmel (1993), since RAD developers constantly interact with customers, technical orientation may be less significant than the need for developers to possess good “people skills.” People skills are no less important within the RAD team itself: individuals must know how to collaborate and compromise to get the job done. According to Jackson and Embly (1996), the goal of the JAD approach is to produce higher quality software in an accelerated time by having both users and analysts accept responsibility and become heavily involved in the system development. As per Carmel et al. (1993), the information systems community subscribes to the notion that the success of a system is proportional to the degree to which the “users” of that system are “involved” in its design and development. According to Keil and Carmel (1995), it has been recognized that customer-developer, mutual understanding and user participation are important factors in the

successful development and implementation of the systems. According to them, the issue that software development managers must grapple with is not *whether* customers should participate in the development process, but *how* they should participate. They suggest to use not only direct relationship but also indirect relations as well. According to Carmel et al. (1993), reported benefits of JAD are time savings, avoidance of cost, completeness and better acceptance of the system by the users. According to Duggan and Thachenkary (2004) a successful JAD process may reduce the communication barriers to effective requirements elicitation and analysis and eventually help to improve the quality of the final system. Further, JAD and its derivatives have become increasingly popular in systems development and other organizational, decision making contexts. It is considered as the best practice for user commitment. JAD is often used with RAD and DSDM. It is considered a risk-reducing investment in ISs development. Therefore, it is clear that number of authors have suggested that JAD is an effective tool in bridging the gap between the user and the modeller in the software development process.

8.5.2.3 Expert Opinions

The next strategy used in validating the proposed technology was to obtain opinion regarding the proposed method from the experts in the field of simulation. Eleven experts, both from the academia and industry expressed their opinion regarding the proposed methodology. The views expressed by them are as follows.

- There is a potential for applying software engineering techniques such as RAD in simulation model development for accelerating the process.

- Typically and frequently the lack of understanding/miscommunication between the user and the modeller lead to delays in the simulation model development process. This is resolved by having the simulationists become the domain expert.
- Simulation software independent conceptual model is a better way of improving the understanding of the system.
- With the development of Visual Interactive Modelling Systems(VIMS) such as Witness, SIMUL8 and Automod there is no clear separation between the conceptual model and the computer model and hence programmers tend to avoid conceptual model development phase and directly move into the computer model
- There is a suspicion among some experts about the feasibility of translating the conceptual model into the computer model.

The need for a conceptual model besides the developments of the simulation software is discussed in detail chapter ten. The feasibility of translating the conceptual model into the computer model was demonstrated by applying the tool for three different cases.

8.6 Summary

A new methodology for accelerating the simulation model development based on RAD approach of software engineering was presented in this chapter. Under this approach it was proposed to accelerate the process by improving the understanding of the system between the user and the modeller and by easing the programming task of the modeller. Three components of the proposed methodology are Joint Application Development Teams, Computer Aided Simulation Model Development tool and Translator programme to

translate the developed conceptual model into the simulation model. The proposed approach will allow the modeller and the user to develop the right model to represent the actual system at a shorter time. This will eventually improve the effectiveness as well as the efficiency of the simulation model development process. Case studies, literature and expert opinions supported the claimed benefits of the proposed methodology. The CASE tool developed under the proposed approach to help to develop the simulation software independent conceptual model and subsequently to translate that into the computer model is explained in detail in the next chapter.

Chapter Nine

Presentation of the Rapid Simulation Model Development Tool

In the previous chapter it was found that it was necessary to have a CASE tool to develop the conceptual model of the system to be simulated and subsequently a program generator to translate the conceptual model into the computer simulation model. A combined tool named Computer Aided Simulation Conceptual Model Developer (CASCoMoD) was developed using Visual Basic, Visio, VBA, MS_Access, MS_Powerpoint, MS_Excel and Arena to achieve these two objectives. The features and technical details of the tool are explained in this chapter.

9.1 Structure of the Tool

In order to develop a simulation model, a user¹ of CASCoMoD should follow four stages;

- Initialization of the system
- Construction of the conceptual model
- Construction of the Quantitative Model
- Construction of the simulation model

Figure 9.1 is the first screen the user sees upon loading CASCoMoD. With the main menu shown in this screen, the user has the opportunity to initialize a new system or to continue the initialization of a saved system created on an earlier occasion. This component of the programme is written by using Visual Basic.

¹ Throughout this chapter the term 'user' is used to denote the user of CASCoMoD, i.e. the modeller and the user of the system to be simulated.

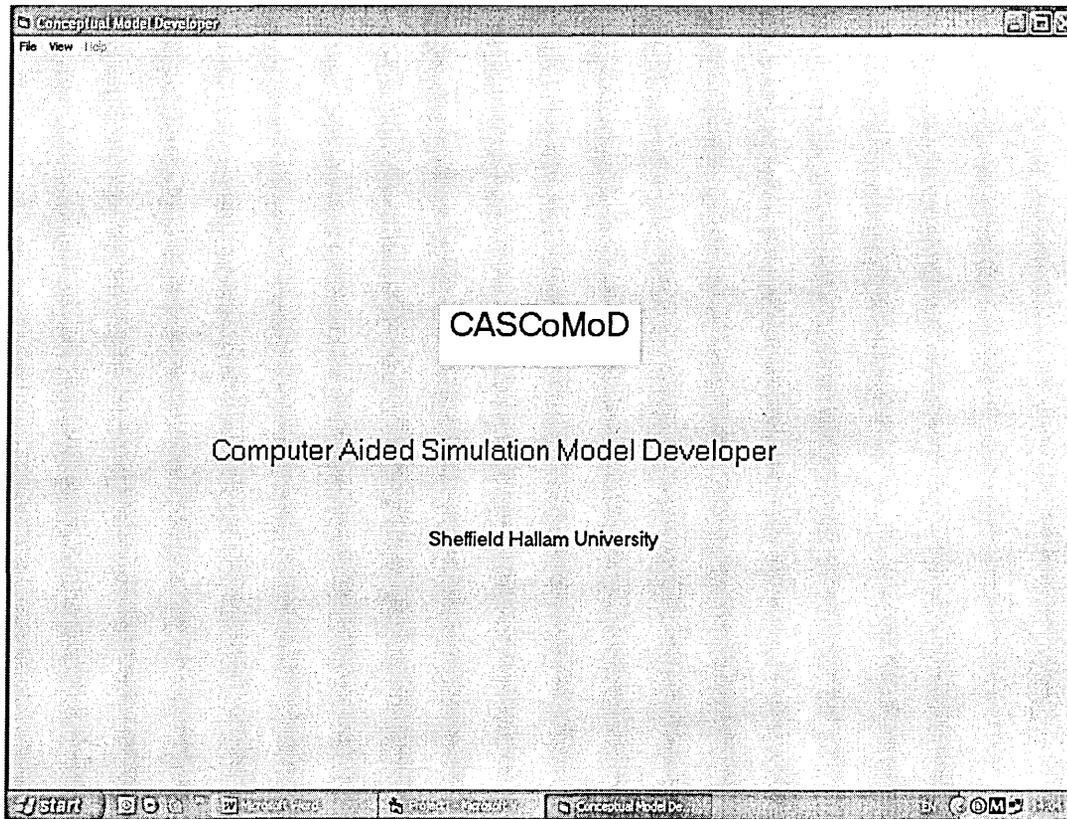


Figure 9.1: Main Menu Screen

9.2 Initialization of the system

Project initialization process consists with three phases. In phase one (see Figure 9.2), the user should provide general information regarding the project, such as company name, address and contact person.

The screenshot shows a 'Project Initialization' dialog box with three tabs: 'General Info', 'Machine and Process Info', and 'System Info'. The 'General Info' tab is selected and contains the following fields:

- Company Name: Small Manufacturing Co.
- Address: Any Street, Any City, Any Country
- Contact Person: A. N. Other
- Telephone: 0114-2253395
- Fax: 0115-2223394
- e-mail: another@anyserver.co.uk

The 'Machine and Process Info' tab contains:

- Model Name: SMLManufac
- Ref. No.: SMLManufac/03/00

At the bottom of the dialog, there are three checkboxes:

- Company Information Saved
- Machine and Process Information Saved
- System Information Saved

Buttons for 'Save', 'Cancel', and 'Quit' are also present.

Figure 9.2: Initialization of the Project-Screen 1 of 3 – General Information

The user can provide machine and process information in the second phase (see Figure 9.3). S/he has the option to select the type of system to be simulated from a library of systems or to create their own system. For example, if the user selects 'Manufacturing' as the type of the system, the names of machines and processes found in a typical manufacturing factory are populated into the right hand side lists of the of the frames below the text box (see Figure 9.3). The user can either select machines in the system from the list of machines in the right hand side list or add new machines. In the same manner the user can provide processes information as well. One of the important features of CASCoMoD is that it has the ability to automatically update the library of systems with the new information provided in this phase.

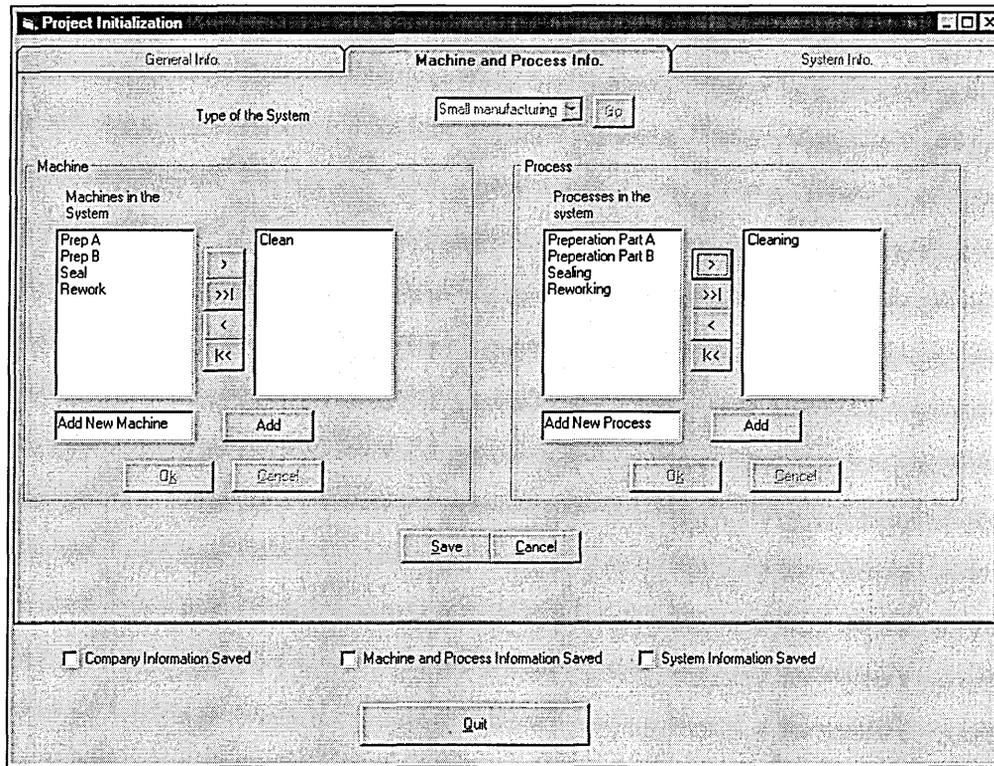


Figure 9.3: Initialization of the project – Screen 2 of 3 – Machine and process information

In the third phase of the initialization process, the user should provide the system information (See Figure 9.4). Three categories of information regarding the system are to be provided in this phase, namely input, process and output.

Project Initialization

General Info. Machine and Process Info. System Info.

Input Process Output

Number of Parts go through the system 2 Ok

Enter the name of part 2 Part B Ok

List of Parts

Part A
Part B

Save Cancel

Company Information Saved Machine and Process Information Saved System Information Saved

Quit

Figure 9.4: Project initialization – Screen 3 of 3 – System information (Inputs)

Under the inputs, number and names of the parts go through the system can be specified (Figure 9.4). Under the process information, the user should select the process or processes which can be performed by each machine defined in the second phase. This information will be useful in identifying any multi-tasking machines in the system, i.e. there may be certain machines which can perform more than one process.

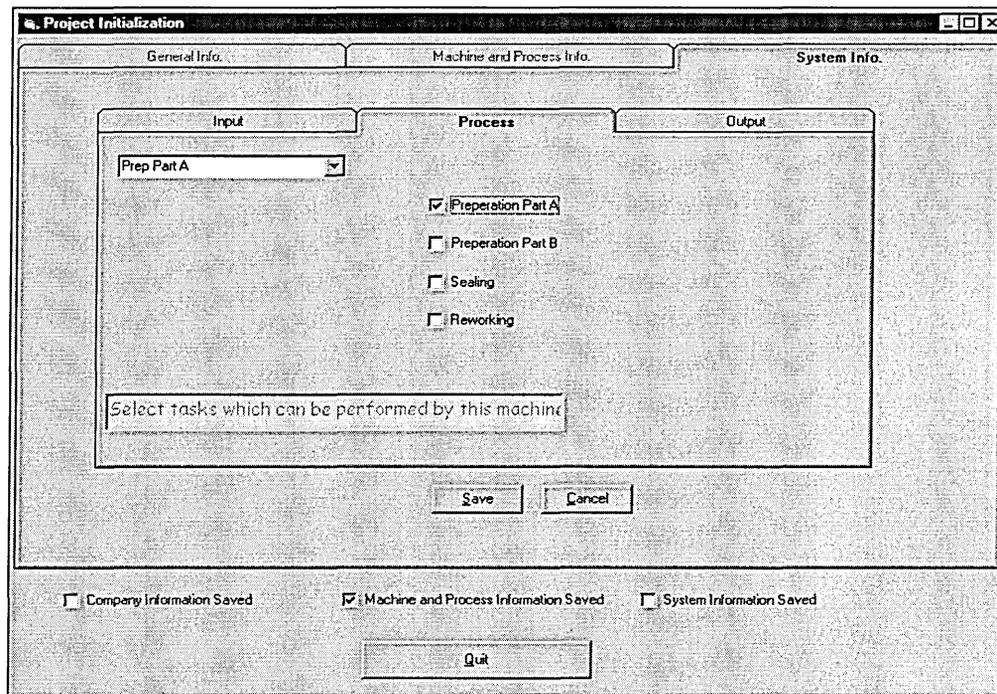


Figure 9.5: Project initialization – Screen 3 of 3 – System information (Processes)

Under the output information the user should specify the number and names of the exit points of each part type (See Figure 9.6).

All those information provided through the initialization process are saved in a text file. This text file is used to extract information needed to build the conceptual model of the system. Once the initialization process is completed the user can select the Conceptual Model option from the main menu in order to develop the conceptual model of the system to be developed.

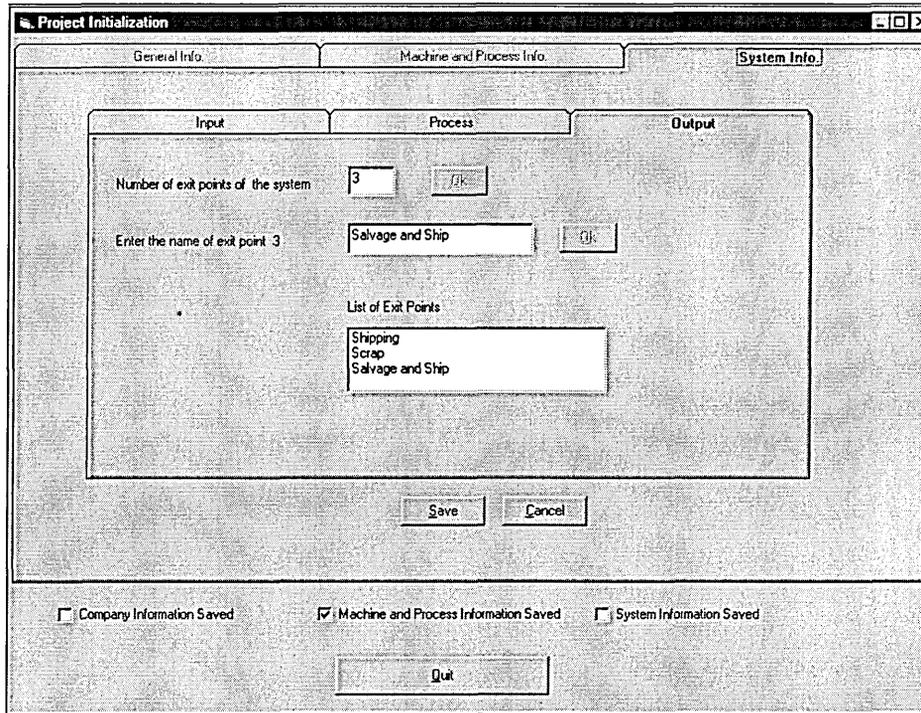


Figure 9.6: Project initialization – Screen 3 of 3– System information(Outputs)

9.3 Development of the Conceptual Model

This component of CASCoMoD was developed using MS_Visio as the drawing package and VBA as the programming language. Selection of the Conceptual Model menu item from the main menu will open the screen shown in Figure 9.7. The user can use the menu item called 'Project' found in the Visio menu bar to develop the conceptual model and subsequently to convert it to a computer simulation model. When the tool is opened in MS_Visio, a few pages, one page each for 'Layout', 'Processes and Exits', 'Complete Model' and one page for each part type through the system to show its route through the system, are automatically displayed on the drawing.

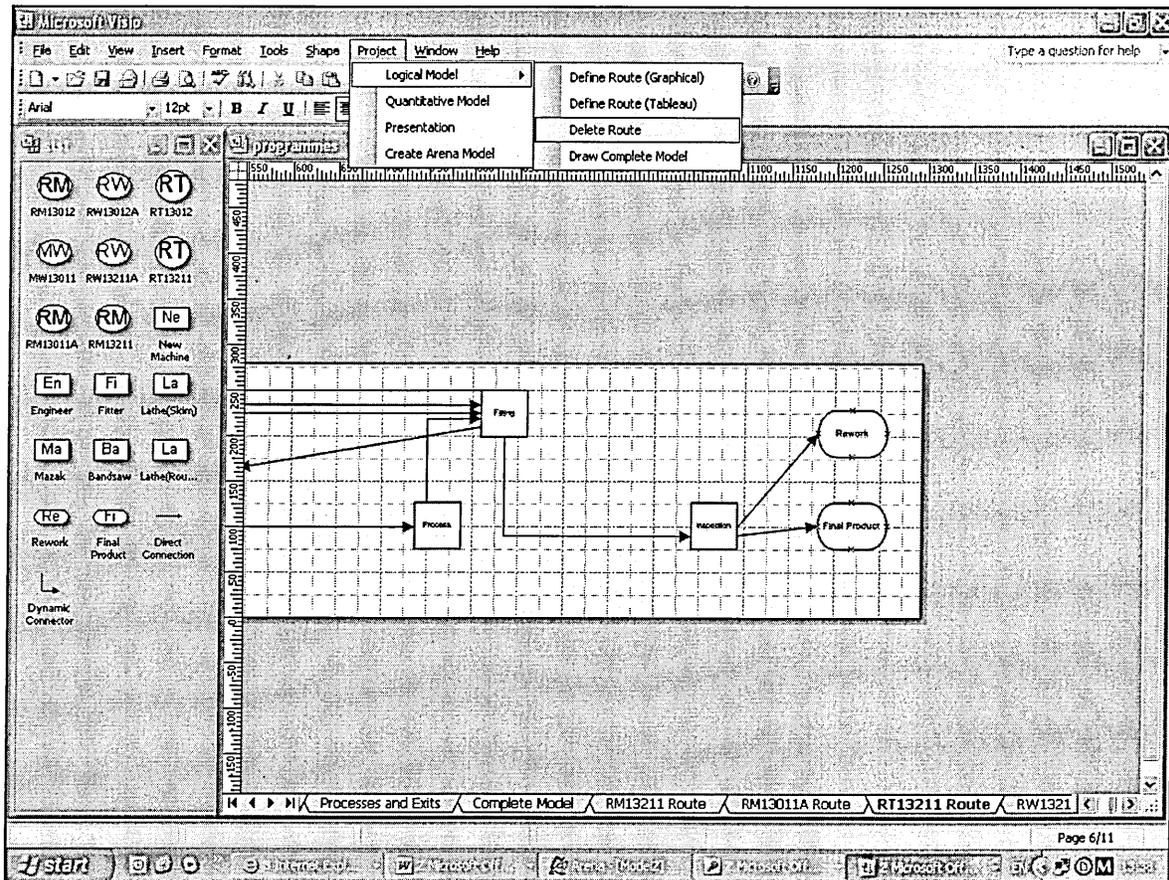


Fig 9.7: Visio Screen with the model stencil and Project Menu

At the beginning, all the pages except 'Process and Exits' page which contains symbols representing processes and exits defined in the initialization phase are empty. In addition to that a stencil containing symbols representing part types, machines, exit points and connecting arrows are also opened in the screen.

Then the user can gradually define the route of each part type by selecting processes and exits involved from the 'Processes and Exits' and thereafter by selecting 'Define Route' from the project menu. Then all the selected processes and exits are copied into relevant

route pages. Thereafter the user can define the route of that particular part by selecting appropriate arrow symbols from the stencil. Or else the user has the option to define the route of a part by identifying 'from machine' to 'to machine' as well². In order to delete a defined route, user should select 'Delete Route' menu option.

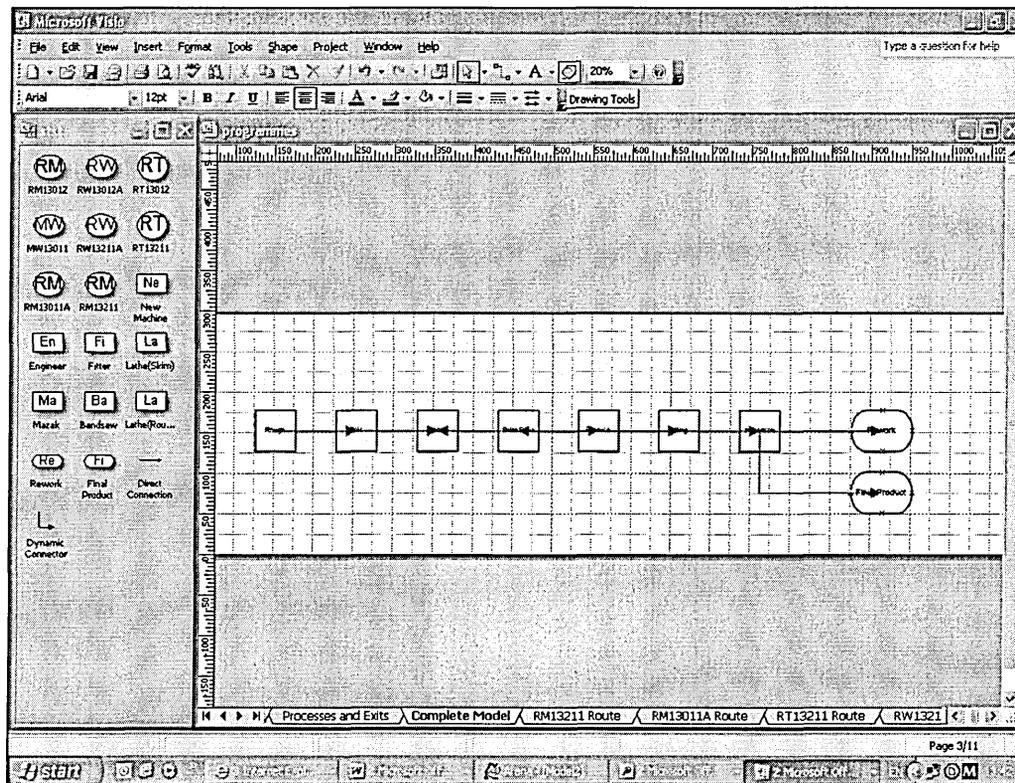


Figure 9.8 : Complete model of the system

Once all the routes are defined, the user can select the 'Draw Complete Model' menu option from the Project menu to build the complete model of the system. This program is capable of identifying connections starts from the same process and ends at the same process/exit but appear in different route pages. For example, both Part A and Part B move

² This option was not available in the earlier version of CASCoMoD. It was included after applying the earlier version into the case study

from Reworking to Scrap. But only one connection will appear in the complete model (Figure 9.8). When all those pages are taken together, that provides a conceptual model or a prototype of the system to be simulated.

9.4 Construction of the Quantitative Model

CASCoMoD provides a facility to include the quantitative model also into the conceptual model through a simple spreadsheet (See Figure 9.9). The programme will create a workbook with a separate worksheet for every part type move through the system. Only the machines which that particular part type moves through are displayed on the relevant worksheet. The user should provide set up time and process time at each machine for each part type.

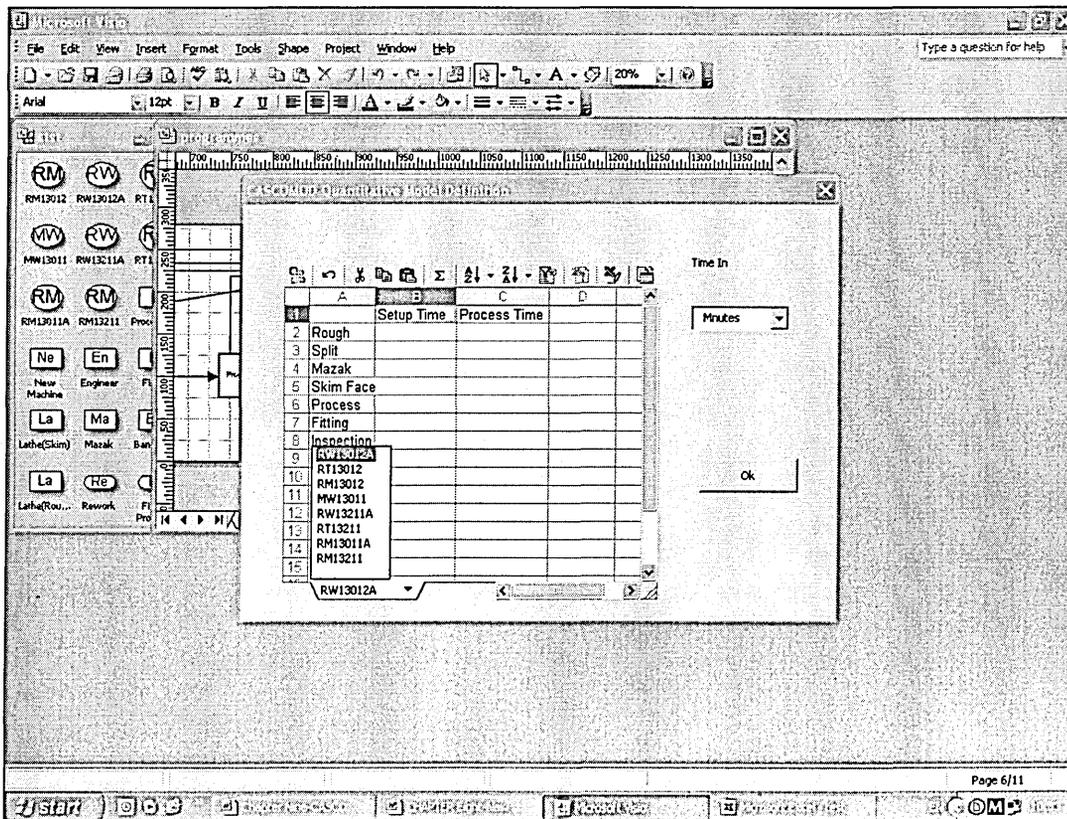


Figure 9.9: Inclusion of the Quantitative Model

9.5 Presentation of the Model

Practically, only one or two representatives from the client organization may be involved in the conceptual model building phase. But as shown in the case studies, another set of persons may involve in the validation of the computer model. Therefore, it is required to obtain consensus of all stakeholders before the computer simulation model is built. If this tool is used by two or three people then the Visio drawing may be sufficient to understand the system. But if the model is to be presented to a larger audience, for example a group consisting with managers and users, Visio drawing may be too small and may not be very attractive. To overcome this problem a facility is provided to create an MS_PowerPoint presentation from the created conceptual model (See Figure 9.10). Each page of the drawing becomes a slide of the presentation. This facility will allow the conceptual model creators to explain and validate the conceptual and to obtain the feed back from a larger audience.

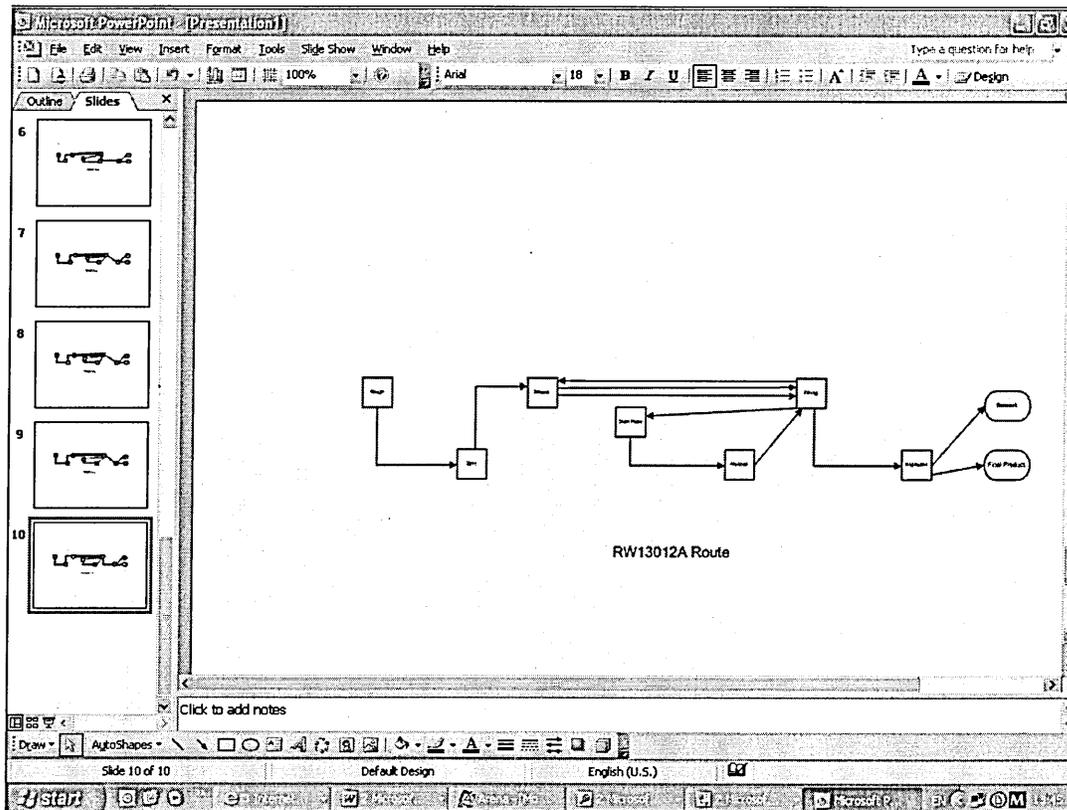


Figure 9.10: PowerPoint presentation of the conceptual model

9.6 Translation of Conceptual Model into Computer Simulation Model

The conceptual model developed through CASCoMoD will accelerate the simulation model development process by elimination or effectively reducing the necessity of rework. However if the built model can be directly translated to the computer model, that will further enhance the acceleration. Therefore CASCoMoD was equipped with a facility for translating the conceptual model into a computer simulation model.

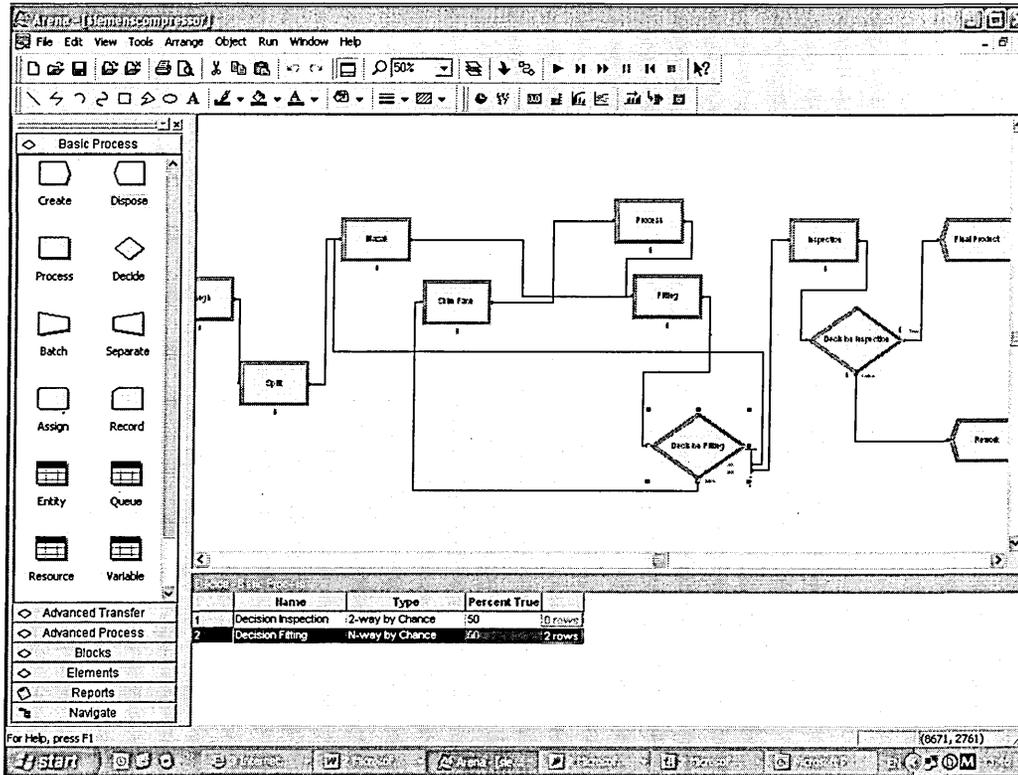


Figure 9.11: Arena Model of the system

Thus far the whole model was built independently of any simulation software. Finally once the complete model is built (i.e. after necessary adjustments are done based on the feedback obtained through consultation of relevant parties), the user can select 'Create Arena Model' from the Project menu to create an Arena simulation model (Figure 9.11) from the conceptual model. As mentioned in chapter six, the results of the questionnaire survey revealed that Arena is the most popular simulation software. Forty one percent of the respondents stated that they use Arena to develop simulation models. Therefore Arena was selected as the simulation software to demonstrate how the conceptual model can be converted into the simulation model. Since the conceptual model is independent of simulation software, if software such as Extend or Promodel is the modeller's choice, then only this particular translation module will have to be written. All other programmes will

remain the same. Import and Export of model facilities provided by Arena were used in translating the conceptual model into the Arena model. When translating the model, the 'complete model' built is used. In the complete conceptual model, all the processes are converted into Process modules and all the exit points are converted into Dispose modules. Create module and an Entity module for each defined part are produced through the programme. If more than one route is available for a process then a Decide module is added next to the particular Process module. Then the modules of the model created by translation programme are Process, Create, Entity, Decide and Dispose, all are from the Basic Process Panel of Arena. Then all these modules and relevant connections are written into an Access database in a format required by Arena. In order to import the model from Access to Arena and to export the model from Arena to Access two DLLs supplied by Arena, namely objImportDLL and objExportDLL are used. Finally the user can use 'Tools->Import Model from Database' option in Arena to open the created model (See Figure 9.11). However the model created is not the final complete model. Since the basic structure and the logic of the computer simulation model is already built, the modeller can make necessary adjustments before delivering the final computer model.

Figure 9.12 shows the relationship between different components of CASCoMoD and software involved in each component.

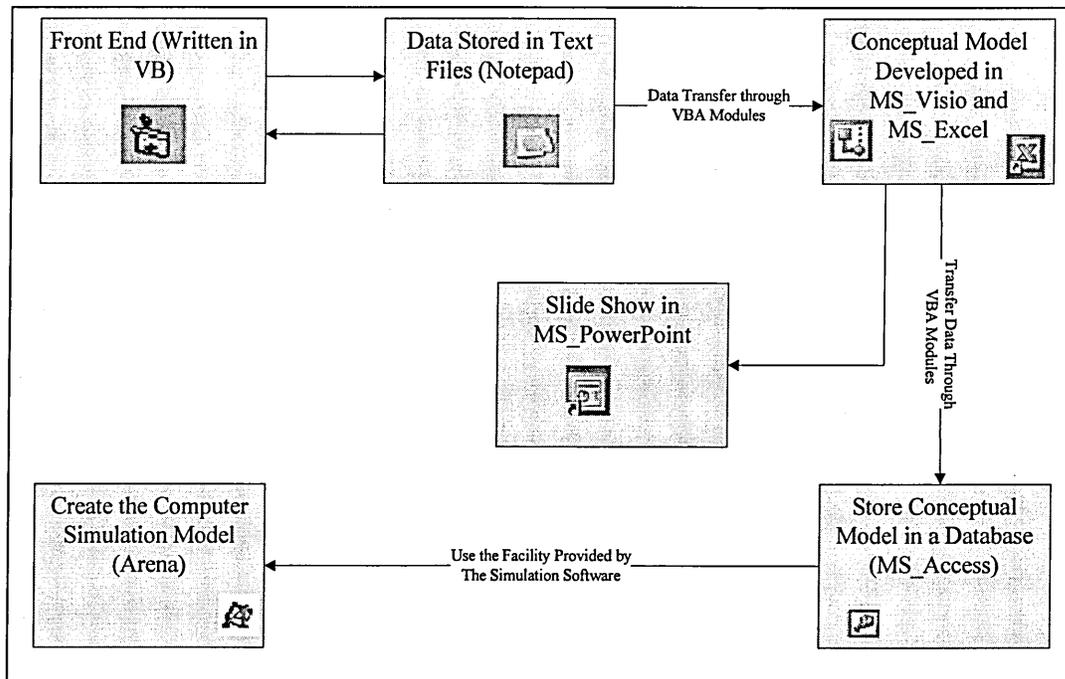


Figure 9.12 : Interrelationship among different components of CASCoMoD.

9.7 System Requirements

Minimum software requirements needed to run the programme are

- Microsoft Visio
- Arena
- Microsoft PowerPoint
- Microsoft Access
- Microsoft Notepad
- Microsoft Windows
- Microsoft Excel

9.8 Summary

The tool developed to support the acceleration of simulation model development process, CASCoMoD, was explained in this chapter. Broadly there are four components in the tool, namely; initialization of the system (a Visual Basic programme), conceptual model development (by using Visio and VBA programme), quantitative model development and the translation of the Visio conceptual model into a simulation model, in this case to an Arena model, by using a VBA programme and through an Access database. Practical implications and further research are discussed in the next chapter.

Chapter Ten

Discussion and Conclusions

This chapter starts with a discussion on implementation issues of the proposed new rapid simulation model development methodology which was described in chapters eight and nine. Then it goes to explain the implications of the new methodology to the simulation industry. Conclusions of the study and further research are given in the latter part of the chapter. It ends with a summary of the research.

10.1 Implementation Issues of the Proposed Methodology

Issues being addressed in implementing the proposed approach are explained in the following sections. One of the major premises of the proposed methodology was development of simulation software independent conceptual model of the system to be simulated. There are various options available for the modeller to develop the conceptual model ranging from paper based method to use of simulation software itself. With all these options why it was necessary to develop a new tool, i.e. CASCoMoD, is discussed below.

10.1.1 Alternative Approaches to Develop the Conceptual Model

As mentioned in the second chapter, practitioners are having the view that many of the pitfalls in the latter stages of the simulation model development process can be avoided by structuring conceptual model before building it on the computer (Robinson,1994; Shannon,2000; Mehta,2000). This view was confirmed by the questionnaire survey results as well. Respondents of the questionnaire survey agreed with the statement “A conceptual model is built before the computer model is built.” According to Robinson (1994), a day on paper saves a month on a computer. As pointed out by Kelton et al. (1998) a lot of

modellers use pencil (a pen for the overconfident) and a paper to sketch out their logic using familiar terminology. Others use a more formal approach and create a logic diagram using standard flowcharting symbols. Therefore, the options available for a simulation modeller to develop a conceptual model are as follows.

- i. Use a pencil and a paper.
- ii. Follow a formal standard such as Flowcharts, IDEF or PetriNet (either paper based or computer based).
- iii. Use a general purpose software such as MS_Word.
- iv. Use a graphic software such as Visio or AutoCAD.
- v. Use a simulation software such as Arena.

The most flexible among those methods may be using the paper and the pencil because it provides the freedom for the modeller to draw whatever comes to his mind. However, it is impossible to translate the drawn model into a simulation model. Therefore, there is no guarantee that the developed computer model is actually a reflection of the conceptual model developed on the paper. Hence, even though it improves the understanding of the system the time spent on development of the conceptual model is a waste from the of model development point of view. Further, using paper and pencil is a very cumbersome process and the final product is not tidy and neat.

As an alternative to the paper based conceptual model development it is possible to use a general word processing software such as MS_Word to develop the conceptual model on a computer screen. The major advantage of the computer based method over the paper based method is that it is less cumbersome and the final output is neater. But the programming

capability of MS_Word is very limited so it is still the developed model cannot be directly translated in to a simulation model.

The problem with formal standards such as flowchart and PetriNet are that they are mainly tools available for modellers. They are not user friendly and difficult for a user who is not familiar with simulation and/or the technique to understand the output. As explained in chapter eight the main feature behind the proposed tool is simplicity.

Another option available for conceptual model development is to use the simulation software itself. With the development of animation facilities of simulators, modellers tend to avoid conceptual model development and straight away start the computer model by using the simulator. One of the issues raised by the experts as mentioned in chapter eight was that isn't it possible to use the simulator itself to develop the conceptual model. But it is not flexible and the simulation software are not intended to develop conceptual model. The basic fact is that simulators are not designed to be drawing packages. But in developing the conceptual model what is important is showing the relationships in a graphical format.

Attribute	Simulation Software	Proposed Tool
Output	Computer Model	Conceptual Model
Objective	Develop the computer model. Display the behaviour of the system as a whole, i.e. for example understand bottlenecks of the system when all the parts go through the system.	Understand the behaviour of the each part type, i.e. how does the each part type move through the system.
Approach	Bottom up	Bottom up
Orientation towards	Programming side of the model	Operational side of the system
Perspective	Modeller perspective	User perspective
View of the System	Holistic	Individual scenarios
Terminology	Key words/Technical terms	Terms from the user domain
Modules	All the modules needed to run the computer model	Essential modules needed to understand the system
Begin the process with	Full description of the system	Some vague idea about the system
Logic of the model (Specifications of the system)	On the hand of the modeller before the modelling processing starts	One of the outputs of the tool
Completeness of the final outcome	Fully complete	May be fully complete/may be just incomplete bits and pieces
Alterations to the outcome	Difficult	Easy

Table 10.1: A comparison between the simulation software and CASCoMoD.

Further, when using simulation software to build the conceptual model, it is quite obvious that simulation terminology and symbols which may not be familiar to an ordinary user are

used. The modeller naturally tends to develop the conceptual model with the Aréna or WITNESS or whatever the simulation software building blocks in mind. The final output may appear to be closer to the simulation model than the system to be simulated. This will help the modeller to build the model quickly but will not serve the purpose of the conceptual model, i.e. understanding of the system to be simulated. Another reason for having a tool for developing the conceptual model is that even experienced modellers may find it difficult to develop a running computer simulation model in front of a user. Sometimes even a simple error may crash the programme and it may take hours to identify the error. This may lead to the loss of confidence of the client. Therefore, it is apparent that even with developments in simulators, there remains a need for a tool to develop the conceptual model. Table 10.1 provides a comparison between the simulation software and CASCoMoD.

It seemed that one of the graphical software such as Visio or AutoCAD was the most appropriate tool for developing a conceptual model. AutoCAD has a lot of facilities for creating technical or complex drawings. However, since it was decided to use only a few basic symbols such as boxes, arrows and circles to develop the conceptual model in order to keep the conceptual model simple and easy to understand, MS_Visio seemed to be the most appropriate graphical software which satisfies the above mentioned requirements. Visual Basic and VBA were selected as programming languages as it was easy to program with them when compared to other languages such as C. From the e-mail questionnaire survey it was found that Arena is the most popular simulation software among respondents. Therefore, in order to demonstrate the capability of translating the conceptual model to the computer simulation model, Arena was selected as the simulation software. The tool was

designed in such a way that only the commonly available software were required to implement it.

10.1.2 Purpose of the Tool

One of the objectives of the CASCoMoD is to improve communication between the user and the modeller and thereby to reduce the time needed for simulation model development. Mathews (2000) has suggested some guidelines for effective presentation of technical information.

- Use graphical methods of communication wherever possible
- Supplement algebraic and mathematical information with geometry to make it simpler and/or clearer
- Use visual models to portray ideas
- Do not be frightened to make approximations where necessary
- Use sketches, diagrams and drawings

Communication involves the transfer of information from one party to another and back again. In the case of simulation, communication is done mainly between two parties i.e. the user and the modeller. Warner (1996) has quoted J. Parry that *“in human communication a great deal of failure comes about not because information has been lost in transmission but because the sender is unable to express what he has to say, or because the receiver is unable to interpret the message in the way intended.”* In simulation, communication failure is inevitable due to the fact that the two parties work in two paradigms. Generally, the user is a domain expert and the modeller is a simulation expert with little or no knowledge of the

other party's operating environment. Therefore it is very important to understand the user's requirements precisely before a simulation model is built. As mentioned in chapter seven, a simulation model is also a piece of computer software. According Insfrán et al. (2002) "quality [of software] means accordance with user requirements. If the resultant software product represents user wishes in an adequate way, and this representation has the required functionality, we have a successful software project." According to them, to go from the problem space to the solution space three main aspects must be considered.

- How to assure the user requirements are captured in a precise way
- How to represent them properly at the problem space level in a correct Conceptual Schema (CS)
- How these conceptual constructs are properly translated into their corresponding software representations at the solution space level, thus obtaining the desired final software product.

In this context, CASCoMoD captures the user requirements through the conceptual model and translates that to computer simulation model. As explained by Kelton et al. (1998) there are two aspects of a simulation model. A modeller may first think of the logical aspects of the model like what the entities and resources are, how entities enter and may be leave the model, the resources they need, the paths they follow and so on. These kind of activities might be called *structural modelling* since they layout the fundamental logic of what you want your model to look like and do. There are other things in specifying a model that are more numerical or mathematical in nature such as the distribution of inter-arrival time, processing time etc. These kinds of specifications might be called *quantitative modelling*. Both of these aspects are equally important in obtaining correct results of a simulation

project. However, the most challenging task is to develop the structural model. Therefore CASCoMoD allows the modeller and the user to capture the structural model first and then to include the quantitative model too into the conceptual model. Even though CASCoMoD is an automated conceptual model development system, it is really a semi-automated system rather than a fully automated system. This assures that the user and the modeller still have the flexibility which they would enjoy if the paper based method was adapted.

According to Howard (1997), prototyping promotes the “five Cs” of successful system development.

- Collaboration – forge a partnership between the users and IT personnel
- Communication – all parties in development project understand each other
- Conceptualization – project members develop visions of solutions which can then be turned into reality
- Change – inevitable consequence of turbulent real world environments
- Consensus in IT – projects have frequently been reached to neither developer’s nor user’s ultimate satisfaction

The conceptual model developed by CASCoMoD also performs the same function. In order to make the communication between the modeller and the relevant stakeholders effective, the conceptual model is developed independent of the simulation software and it is developed in the user’s domain. Consensus from all the relevant stakeholders is obtained by doing PowerPoint presentations of the conceptual model to them and by obtaining their views. Only a few basic symbols such as a box, oval and arrow are used to develop the conceptual model. This allows the modeller to concentrate only on the conceptual model rather than the final simulation model. From the user’s point of view, s/he does not have to

be conversant with simulation terminology or symbols. The modeller and the user will collectively develop a conceptual model using the language and terminology of the users' application domain. It will be closer to the system to be simulated than to the computer simulation model. The main objective of the conceptual model is to specify the characteristics of the proposed system from the user's perspective.

According to Pidd (1996), models should be developed gradually. In the case of CASCoMoD, it allows the user and the modeller to breakdown the model into small components, i.e. rather than considering the all routes of all entities through the whole system. This possibility of breaking down into scenarios allows the participants to understand the system properly.

The other objective of CASCoMoD, in addition to the improvement of communication between the user and the modeller, is the translation of the conceptual model into the computer simulation model. This will ensure that the time spent on development of conceptual model is not wasted. There were some programming problems in this stage. For example, serial numbers of modules in Arena cannot be controlled by the programmer. They are assigned by the Arena through an internal procedure. Therefore, it was necessary to use some ineffective programming techniques which led to the delay in the process. However, CASCoMoD is capable of using every piece of information used in the development of conceptual model in development of the computer model.

Even though, it was not a major objective of CASCoMoD, it provides a facility to document the conceptual model. The final outcome of CASCoMoD will be a document of the logical model in graphical form. This will be helpful for future developers of the model

as a good reference material. Under the traditional approach, model building is started with a written specification of the system. But with the new tool, it is one of the outcomes of the tool.

10.1.3 Joint Application Development Approach

The other major component of the proposed methodology is the adaptation of Joint Application Development Approach in simulation model development. According to Howard (1997), traditional development is based largely on written communication. In contrast, prototyping relies much more heavily on interpersonal skills and verbal communication. Personal relationships between developers and users are more important. Mutual trust, understanding, confidence and commitment are vital in prototyping situations. According to Kosman (1997), involving the user community during the prototyping phase can identify required functionality, eliminate unnecessary functionality, correct erroneous processing, and simplify the GUI before any specification is written or production code is developed. After reviewing a number of articles, Pendergast et al. (1999) identified several advantages of group system development if key stakeholders are involved. First, the overall functional requirements are aired early in the process, making all participants aware of the collective view on the functionality of the new system. Second, system requirements are defined much more quickly than when users are interviewed serially. Third, conflicts that participants cannot resolve among themselves can be brought to the top management attention by the team leader. Fourth, a requirements document resulting from collaboration work is less likely to be challenged later in the project life cycle than one prepared exclusively by an analyst. Therefore, in order to enhance the advantages of developing the

conceptual model/prototype of the system to be simulated, it was proposed to carry out the model development by a small team consisting of the modeller, the user and any other stakeholders of the system. Even under the present system the user is involved in the model development process. But he is not actively and continuously engaged in the model development process. In the other approaches for accelerating the simulation model development process which were explained in chapter five, either the modeller or the user were mainly responsible for the development of the model. Under the proposed approach, in contrast to other approaches, neither the modeller nor the user is solely responsible for the development of the model, but both are equally responsible. Participation of the user is important because s/he is the domain expert, i.e. who knows the operational activities of the system and participation of modeller is also equally important because s/he is the simulation expert, i.e. who knows simulation and simulation software. Therefore the active and continuous participation of both parties is essential for the successfulness of the simulation project. It is necessary that both the modeller and the user to talk and listen to each other. A model representing the actual system can be built only through the efficient and effective dialog between the user and the modeller.

It may be necessary to do some team building exercises to improve the interaction between users and modellers who are from different organizations. It is possible to improve the productivity of the process by providing 'clean room' for the team to work, i.e. release them from other day to day duties to focus their attention on the model development, at the factory/office where the actual operations take place. With the help of a laptop computer, the team can walk through the factory, in the case of an existing factory, and develop the conceptual model by obtaining more information from the people at the factory floor.

10.2 Implications to the Simulation Industry

It was identified earlier that if the conceptual model can be directly translated into the computer simulation model it will substantially accelerate the simulation model development process. The feasibility of the application of the methodology was demonstrated by using Visio, VB, VBA, Access, PowerPoint, MS_Excel and Arena. However, in practice the conceptual model development tool should become a part of the commercial simulation software. But at the same time, it is necessary to keep simulation model and the conceptual model separate from each other and to develop the conceptual model in the user's domain in order to assure that the users of the system are not stranded in the simulation jungle. There are a number of options available for vendors to develop a commercial simulation conceptual model developer:

The first option is to develop a separate module attached to the simulation software for developing the conceptual model, similar to a module such as Input Analyzer of Arena, by the simulation software manufacturing company. So this may be an optional module available for buyers to purchase with the simulation software. This may not be a difficult task because the software developers have access to the source code of the simulation software.

The second option may be that a third party may produce a conceptual model developer and relevant drive programmes to translate the conceptual model into the computer simulation model. Then the buyers can buy the module with the appropriate driver depending on the

simulation software that they are using for computer simulation model building (See Figure 10.3).

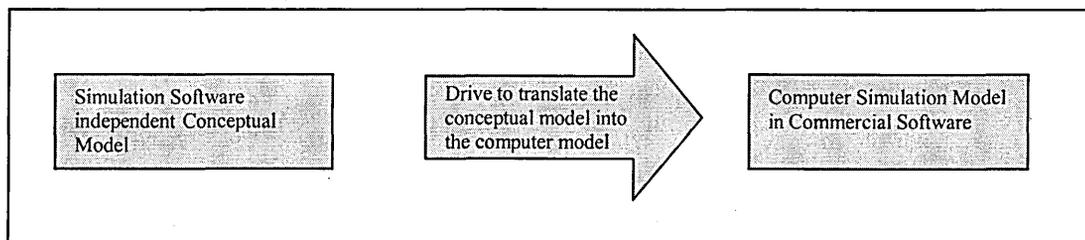


Figure 10.3: Conversion of conceptual model to the computer model through a drive

According to a survey done by *OR/MS Today* journal there are more than 45 Commercial Simulation languages in the current market (Swain, 2003). Ideally, there should be an industry standard to store models into a database. This ideal situation is shown in the Figure 10.4. As noted by Schriber and Brunner (2003), although discrete-event simulation languages are similar in broad terms, they can and typically do differ in subtle but important particulars.

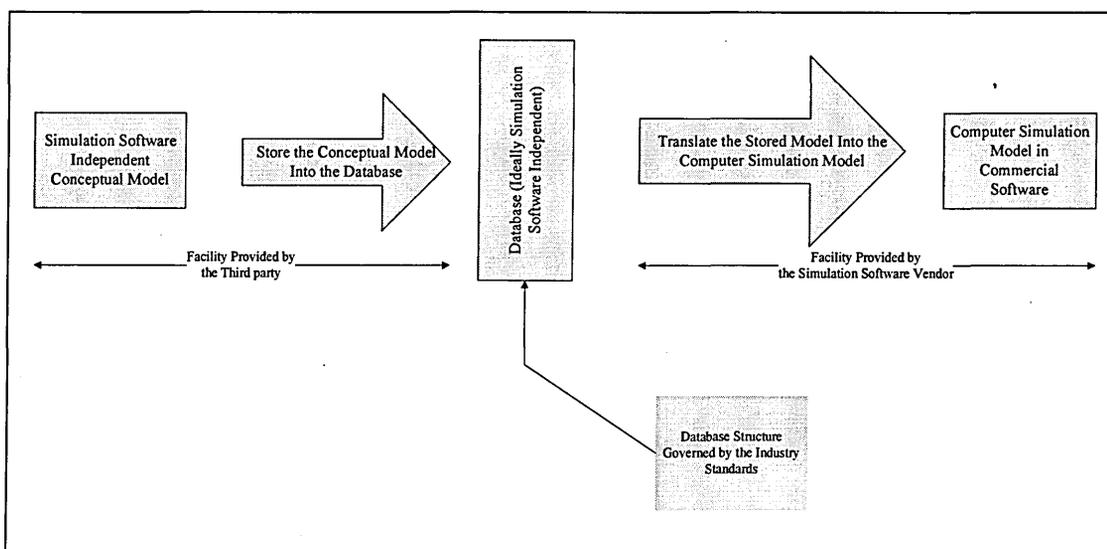


Figure 10.4: Conversion of conceptual model to the computer model through a common database

Even though this sounds like a good idea, there may be number of technical and commercial issues to overcome which may take years. However, it is an issue which should be addressed by the simulation software industry in order to provide more flexibility and interoperability among different simulation software to the simulation community. Even now commercial simulation software provides facilities to store models into a database and extract a model from a database. However, still the database is simulation software dependent, i.e. different simulation software use different database formats. Therefore the third party developing the conceptual model developer has to develop interface programmes for different simulation software. As explained earlier, if there is a common database the same interface could be used for all the simulation software. This would substantially reduce the research and development cost of software and if the cost advantage is passed to the customer, then customers will be able to buy the software at a lower cost. In addition to providing the facility to translate the conceptual model into the computer model it provides other advantages for simulationists as well. For example, a common database will allow the modellers to build and run their models across different commercial simulation software by selecting the appropriate one. Another advantage may be the possibility of using existing available data and information in different formats such as spreadsheets and databases in model development and run through a common database.

The next section presents the conclusions of the research.

10.3 Conclusions

The conclusions of the research are as follows.

- Model development is the most time consuming phase of a simulation project.
- Two major reasons for lengthening the model development process are the lack of understanding of the system to be simulated between the user, the domain expert, and the modeller, the simulation expert, and the difficulty of programming.
- Contemporary researches into accelerating the model development process mainly focus on improving the programming efficiency. Therefore there is a need for research on improving effectiveness of the process, i.e. development of the model required by the client once.
- Present practice of model development closely follows the waterfall approach to software development.
- Simulation model development process can be accelerated by applying Rapid Application Development techniques of software development in simulation model development.
- In the commercial world of simulation software, the conceptual model developer could be a separate module which the users could buy.
- It is beneficial to have an industry standard for a common database to store simulation models which will enhance the interoperability among different simulation software.

10.4 Contribution to the Knowledge

This research proposed a new methodology for accelerating the simulation model development based on improving not only the efficiency but also the effectiveness of the process. The proposed approach explained how the principles and techniques of Rapid

Application Development of software development can be used in the field of simulation model development. The tool developed during the research, CASCoMoD, demonstrated the feasibility of using RAD techniques in the field of simulation. Questionnaire survey done during research revealed the requirements of the modellers and the present situation in terms of simulation model development and it confirmed the findings of the researches done the 1980s and 1990s that the model development is the most time consuming phase of the simulation project life cycle. This is an indication that conducting research on rapid simulation model development remains necessary. The other component of the research strategy, case study research, revealed reasons for lengthening the model development process.

10.5 Further Research

One of the opportunities which was not explored under the proposed methodology is use of reverse engineering for building the conceptual model from a built computer simulation model. Consider a situation where even with all the consultation with relevant stakeholders, it was found that certain amendments to the structural model is needed at a later stage of the simulation model development process. Presently, CASCoMoD is incapable of updating the conceptual model with the changes made to the computer simulation model. What can be done is to make the necessary modifications to the conceptual model and transform it into the computer model. However, all of the additions to the structural model and quantitative model of the computer simulation model made since the last transformation are lost during this process. Therefore to avoid that kind of situation and to provide the flexibility for the model developers to go backward as well, it would be useful to embed

reverse engineering capability also into CASCoMoD. The extended version of the ideal system is shown in Figure 10.5. This structure would allow amendments to be made to the conceptual model or computer model at any stage of simulation model development process. Even now commercial simulation software provides facilities to store models into a database and extract a model from a database. However, still the database is simulation software dependent, i.e. different simulation software use different database formats. Therefore the third party developing the conceptual model developer has to develop interface programmes for different simulation software. As explained earlier, if there is a common database the same interface could be used for all the simulation software. This would substantially reduce the research and development cost of software and if the cost advantage is passed to the customer, then customers will be able to purchase the software at a lower cost.

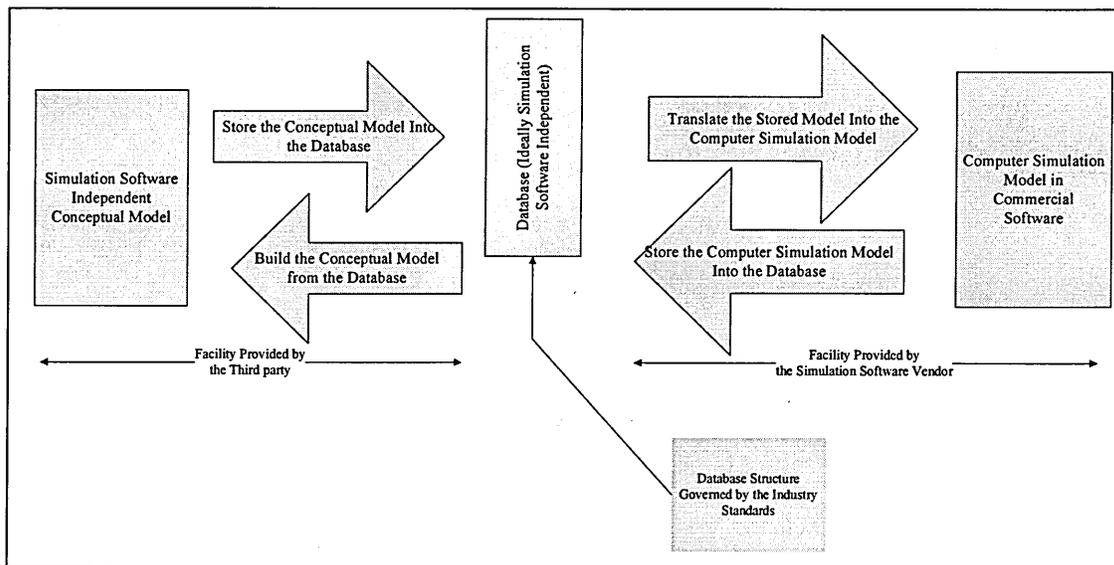


Figure 10.5: The structure of the complete tool for translating the conceptual model to the computer model and vice versa.

Another possible extension of the present research is to develop an internet based conceptual model developer. Under the proposed methodology, the user and the modeller work together (physically) in model building process. Even though the best option will be that the modeller and the user physically work together, in today's competitive business environment modeller's time as well as user's time are expensive. But with the help of the internet it is possible to develop the conceptual model of the system to be simulated while modeller and the user are in different geographical locations. If CASCoMoD can be developed to provide facility to help the modeller and the user to log onto the internet and develop the conceptual model, then both geographical and time barriers are removed and the simulation model development process can be further accelerated. This kind of a system will allow large organizations to have central simulation departments with highly skilled modellers and to provide their service to the users in different places all over the world.

As mentioned earlier, CASCoMoD is a semi automatic system which requires the active participation of both user and the modeller. Yet, it is not proposed to extend research to develop a fully-automated model developer because simulation is both art and science (Shannon,1998; Banks,2000). Future research should be along the lines of providing more flexibility on conceptual model development and easy translation of the conceptual model to the computer model and vice versa (forward and reverse engineering).

10.6 Summary of the Research

Discrete-event simulation is one of the most important tools available to engineers as well as to managers to analyze the behaviour of large and complex systems. It is a multi-stage process and it has been divided into different stages by different authors in different ways. However, the most important stages are problem identification, model building, experimentation and implementation. Contemporary literature as well as the questionnaire survey done among simulation practitioners and academics showed that the model development is the longest among stages of a simulation project life cycle.

As in the case of the simulation project life cycle, the model building process has also been divided into different stages by different authors. However, the three major stages are conceptual model building, computer model building and validation and verification. Literature review and the case study research findings revealed that the two major reasons for lengthening the simulation model development process are the lack of understanding of the system to be simulated between the user, the domain expert, and the modeller, the simulation expert, and the difficulty of programming. Various researchers have proposed different approaches to accelerate the simulation model development process. Research attempts under Model Reuse/Library Driven, Model Reuse/Library Driven plus Knowledge Based, Model Reuse/Object Oriented, Integration with Other Packages and Knowledge Based/Artificial Intelligence approaches are found in the literature. Review of the literature revealed that the majority of researchers have tried to accelerate the model development process by improving programming efficiency. But there is no point of developing a model in a short period of time if it does not represent the actual system to be simulated. Therefore, the effectiveness of the process, i.e. to create a model which representing the

actual system, is crucial for the successfulness of the simulation model development process. With this background it was decided to develop a new methodology for accelerating simulation model development process by adapting tools and techniques of Rapid Application Development methodology of general software development. The most important differences between RAD and traditional approaches are the high user involvement and focus on the duration of the project development over the functionality in the latter case.

Three main components of the proposed methodology are joint application development team, a CASE tool to develop a simulation software independent conceptual model of the system to be simulated, and a programme to translate the conceptual model into the computer model. A tool named CASCoMoD (Computer Aided Simulation Conceptual Model Developer) was built by using VB, VBA, MS_Visio, MS_Access, Powerpoint, Excel and Arena to help the modeller and the user, who work together as a team during model development, to create the conceptual model of the system to be simulated and subsequently translate that into the computer simulation model.

The proposed methodology achieves the objective of accelerating simulation model development process in different ways. Firstly it improves the effectiveness of the process by improving the understanding of the system between the user and the modeller by developing a simulation software independent conceptual model of the system to be simulated. Then the efficiency is increased by solving problems arisen during the model development quickly, reducing the actual programming tasks, reducing the number of programming errors and by reducing time needed to validate the model.

CASCoMoD was developed using commonly available software. However, in the field of commercial simulation software, it is proposed to amalgamate a conceptual model developer into commercial simulation software as a separate module. Another option is to develop such tools with translators (driver programmes for different simulation software) by third parties. In order to make this task easier and to enhance the interoperability among different simulation software, it is proposed to develop industry standards for a common database structure to store models.

CASCoMoD was mainly for forward engineering, i.e. develop the computer simulation model from the conceptual model. However, a possible extension of the research is to provide reverse engineering facility as well to update the conceptual model with the changes done to the developed computer simulation model. Another possibility is to extend the tool to provide facilities to develop the conceptual model through virtual meetings via internet and thereby to avoid the need for the modeller and the user to physically work together but while having the advantages of joint application development approach. Therefore, it is apparent that there is a potential for more research on rapid simulation model development process.

References

- Agarwal R., Prasad J., Tanniru M. and Lynch J., 2000, Risks of rapid application development, *Communications of the ACM*, 43(11), 177-188
- Aggarwal R. and Rezaee Z., 1996, Total quality management for bridging the expectation gap in systems development, *International Journal of Project Management*, 14(2), 115-120
- Alvarez F.J., Israel R.M., and Weitzenfeld R., 2000, Method for the Rapid Application Development, *Proceedings of the International conference on advances in infrastructure for electronic business, science, and education on the internet*, L'Aquila, Italy
- Amico V., Bruzzone A. and Guha R., 2000, Critical issues in simulation, *Proceedings of the 2000 summer simulation conference*, Vancouver, Canada, 893-898
- Andriole S.J., 1991, *Rapid application prototyping*, QED information sciences, Wellesley, MA, Second Edition
- Anglani A., Grieco A., Pacella M. and Tolio T., 2002, Object-oriented modeling and simulation of flexible manufacturing systems: a rule-based procedure, *Simulation modelling practice and theory*, 10(3-4), 209-234
- Arons, H. de S., 1999, Knowledge-Based Modeling of Discrete-Event Simulation Systems, *Proceedings of 1999 winter simulation conference* Eds. Farington P.A., Nembhard H.B., Sturrock D.T. and Evans G. W., Phoenix, Arizona, 591-597
- Arons H. de S. and Asperen E.V., 2000, Computer assistance for model definition, *Proceedings of 2000 winter simulation conference* Eds. Jones J.A., Barton R.R., Kang K. and Fishwick P.A., Orlando, Florida, 241-245
- Banker R.D. and Kauffman R.J., 1991, Reuse and productivity in integrated computer-aided software engineering: an empirical study, *MIS Quarterly*, 15(3), 375-401
- Banks J., 2000(a), Introduction to simulation, *Proceedings of 2000 winter simulation conference* Eds. Jones J.A., Barton R.R., Kang K. and Fishwick P.A., Orlando, Florida, 9-16
- Banks J., 2000(b), Simulation in the future, *Proceedings of 2000 winter Simulation conference* Eds. Jones J.A., Barton R.R., Kang K. and Fishwick P.A., Orlando, Florida, 1568-1576
- Barrow P.D.M. and Mayhew P.J., 2000, Investigating principles of stakeholder evaluation in a modern IS development approach, *Journal of systems and software*, 52(2-3), 95-103
- Basili V.R., 1996, The role of experimentation in software engineering: past, current and future, *Proceedings of 18th International conference on software engineering*, Berlin, Germany, 442 - 449

References

- Benjamin P., Delen D., Mayer R., and O'Brien T., 2000, A model-based approach for component simulation development, *Proceedings of 2000 winter simulation conference*, Eds. Jones J.A., Barton R.R., Kang K. and Fishwick P.A., Orlando, Florida, 1831-1839
- Berends P. and Romme G., 1999, Simulation as a research tool in management studies, *European Management Journal*, 17(6), 576-583
- Beynon-Davies P. Carne C., Mackay H. and Tudhope D., 1999, Rapid application development (RAD): an empirical review, *European journal of information systems*, 8(3), 211-223
- Beynon-Davies P. and Holmes S., 2002, Design breakdowns, scenarios and rapid application development, *Information and software technology*, 44(10), 579-592
- Brooks Jr. F.P., 1996, The computer scientist as toolsmith II, *Communications of the ACM*, 39(3), 61-68
- Brown N. and Powers S., 2000, Simulation in a box - a generic reusable maintenance model, *Proceedings of 2000 winter simulation conference*, Eds. Jones J.A., Barton R. R., Kang K. and Fishwick P.A., Orlando, Florida, 1050 - 1056
- Carmel E., 1995, Does RAD live upto the hype-counterpoint, *IEEE Software*, 12(5), 24-26
- Carmel E., Whitaker R.D. and George. J.F., 1993, PD and joint application design: a transatlantic comparison, *Communications of the ACM*, 36(6), 40-48.
- Carr, M.J., 1989, A circular model for software development, *Proceedings of the sixth Washington symposium on Ada*, McLean, Virginia, United States, 129-133
- Carter C.J., 1997, Rapid integration and testing of business solutions, *BT Technology Journal*, 15(3), 37-47
- Centeno M.A. and Standridge C.R., 1992, Database and artificial intelligence: enabling technologies for simulation modeling, *Proceedings of 1992 winter simulation conference* Eds. Swain J. J., Goldsman D., Crain R. C. and Wilson J. R., Arlington, USA, 181-188
- Cepura N. and Verner J., 1996, Prototyping: some new results, *Information and software Technology*, 38(12), 743-755
- Chun H.W., 1997, Automatic simulation program synthesis using a knowledge-based approach , *Simulation Practice and Theory*, 5(6), 473-488
- Crapo A.W., Waisel A.B., Wallace W.A. and Willemain T.R., 2000, Visualization and the process of modelling: a cognitive-theoretic view, *Proceedings of the sixth ACM SIGKDD international conference on knowledge discovery and data mining*, Boston, Massachusetts, United States, 218-226

References

- Denning P. J., 2005, Is computer science science, *Communications of the ACM*, 48(4), 27-31
- Denning P.J., Comer D.E., Gries D., Mulder M.C., Tucker A. B., Turner A. J. and Young P.R., 1988, Computing as a discipline: preliminary report of the ACM task force on the core of computer science, *Proceedings of the nineteenth SIGCSE technical symposium on Computer science education*, Atlanta, Georgia, United States, 41-41
- Drucker P.F., 1992, *Management*, Butterworth Heinemann, Oxford
- DSDM Consortium, 2002, *DSDM Consortium Manual 4.1: Evaluation Copy*
- Duggan E.W. and Thachenkary C.S., 2004, Integrating nominal group technique and joint application development for improved systems requirements determination, *Information & Management*, 41(4), 399-411
- Fairley R.E. and Thayer R.H., 1997, The concept of operations: The bridge from operational requirements to technical specifications, *Annals of Software Engineering*, 3, 417-432
- Gordon V.S. and Bieman J.M., 1995, Rapid prototyping: lessons learned, *IEEE Software*, 12(1), 85-95
- Hartmanis J., 1995, Turing award lecture: On computational complexity and the nature of Computer Science, *ACM Computing Surveys*, 27(1), 7-16
- Hlupic V., 2000, Simulation software: An operational research society survey of academic and industrial users, *Proceedings of 2000 winter simulation conference* Eds. Jones J.A., Barton R. R., Kang K. and Fishwick P.A., Orlando, Florida, 1568-1576
- Hlupic V., 1999, Simulation software: User's requirements, *Computers and Industrial Engineering*, 37(1-2), 185-188
- Howard A., 1997, A new RAD-based approach to commercial information systems development: The dynamic system development method, *Industrial Management and Data Systems*, 97(5), 175-177
- Howard A., 2000, Rapid application development: Rough and dirty or value-for-money engineering?, *Communications of the ACM*, 45(10), 27-29
- Huang R., 1998, Making active CASE tools-Toward the next generation CASE tools, *ACM SIGSOFT Software Engineering Notes*, 23(1), 47-50
- Insfrán E., Pelechano V. and Pastor O., 2002, Conceptual modeling in the eXtreme, *Information and software technology*, 44(11), 659-669
- Integrated Manufacturing Technology Initiative, 2000, Integrated manufacturing technology roadmapping project modeling and simulation, <http://www.IMTI21.org>

References

- Jackson R.B. and Embly D. W., 1996, Using joint application design to develop readable formal specifications, *Information and software technology*, 38(10), 615-631
- Jain S., 1999, Simulation in the next millennium, *Proceedings of 1999 winter simulation conference*, Eds. Farington P. A., Nembhard H.B., Sturrock D.T. and Evans G.W., Phoenix, Arizona, 1478-1484
- Johnson T., 1999, Building data-driven template model, *Proceedings of the 1999 AutoMod Symposium*, Salt Lake City, UT
- Kamada T. and Kawai S., 1991, A general framework for visualizing abstract objects and relations, *ACM transactions of graphics*, 10(1), 1-39
- Keil M. and Carmel E., 1995, Customer-developer links, *Communications of the ACM*, 38(5), 33-39
- Kelton W.D., Sadowski R.P. and Sadowski D.A., 1998, *Simulation with Arena*, Mc-Graw Hill, Singapore, International Editions
- Khan M.R.R., 1999, Simulation modeling of a garment production system using a spreadsheet to minimize production cost, *International journal of clothing science and technology*, 11(5), 287-299
- Khaznchi D. and Munkvold B.E., 2000, Is information systems a science? An inquiry into the nature of the information systems discipline, *The DATA BASE for advances in information systems*, 31(3), 24-42
- Koh K.H., de Souza R., Ho N., 1996, Database driven simulation/simulation-based scheduling of a job shop, *Simulation practice and theory*, 4(1), 31-45
- Kosman R.J., 1997, A two-step methodology to reduce requirement defects, *Annals of software engineering*, 3, 477- 494
- Kovács G. L., Kopácsi S., Nacsa J., Haidegger G. and Groumpos P. , 1999, Application of software reuse and object-oriented methodologies for the modelling and control of manufacturing systems, *Computers in industry*, 39(3), 177-189
- Lending D. and Chervany N.L., 1998, CASE tools: understanding the reasons for non-use, *ACM SIGCPR Computer personnel*, 19(2), 13-26
- Martin J., 1991, *Rapid application development*, McMillan Publishing, New York
- Mathews C., 2002, *A Guide to presenting technical information effective graphic communication*, Professional engineering publishing, London
- McConnell S., 2000, 10 best influences on software engineering, *IEEE Software*, 17(1), 10-17

References

McMurtrey M.E., Teng J.T.C., Grover V. and Kher H.V., 2000, Current utilization of CASE technology: lessons from the field, *Industrial Management and Data Systems*, 100(1), 22-30

Mehta A., 2000, Smart modeling-basic methodology and advanced tools, *Proceedings of 2000 winter simulation conference*, Eds. Jones J.A., Barton R. R., Kang K. and Fishwick P.A., Orlando, Florida, 241-245

Merkuryeva G.V. and Merkuryev Y.A., 1994, Knowledge based simulation systems – A review, *Simulation*, 62(2), 74-89

Merriam-Webster Online, <http://www.m-w.com> (Accessed 14th April 2005)

Mertins K, Rabe M. and Jaekel F.W., 2000, Neutral template libraries for efficient distributed simulation within a manufacturing system engineering platform, *Proceedings of 2000 Winter Simulation Conference*, Eds. Jones J.A., Barton R.R., Kang K. and Fishwick P.A., Orlando, Florida, 1549-1557

Miller S. and Pegden D., 2000, Introduction to manufacturing simulation, *Proceedings of 2000 Winter Simulation Conference*, Eds. Jones J.A., Barton R.R., Kang K. and Fishwick P.A., Orlando, Florida, 63-66

Murray K.J. and Sheppard S.V., 1988, Knowledge-based simulation model specification, *Simulation*, 50(3), 112-119

National Institute of Standard and Technology, 2000, Manufacturing simulation and visualization, <http://www.mel.nist.gov/proj/msv.htm>

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

Oakshott L., 1997, *Business modeling and simulation*, Pitman Publishing, London

Patton M.Q., 1990, *Qualitative Evaluation and Research Methods*, Second Edition, Sage Publications, London

Paul R.J. and Taylor S.J.E., 2002, What use is model reuse: is there a crook at the end of the rainbow, *Proceedings of 2002 Winter Simulation Conference*, Ed Yücesan E., Chen C.H., Snowdon J.L., and Charnes J.M., San Diego, California, USA, 648-652

Pidd M., 1992, *Computer simulation in management science*, John Willie, Chichester

Pidd M., 1996, Five simple principles of modelling, *Proceedings of 1996 Winter Simulation Conference*, Eds. Charnes J.M., Morrice D.J., Brunner D. and Swain J.J., 721-728.

Pitts G. and Hwang S.P., 1999, An intelligent interface agent: simulation/modeling made simple, *Proceedings of 1999 Summer Computer Simulation Conference*, Chicago, Illinois

References

- Post G., Kegan A. and Keim R.T., 1998, A comparative evaluation of CASE tools, *Journal of Systems and Software*, 44(2), 87-96
- Praehofer H., 1996, Object oriented, modular hierarchical simulation modeling: towards reuse of simulation code, *EUROSIM – Simulation News Europe*, 17, 5-8
- Pressman R.S., 2000, *Software Engineering, A Practitioner's Approach: European Adaptation by Darrel Ince*, McGraw Hill, Berkshire
- Rathburn T.A. and Weinroth G.J., 1991, Desktop simulation : modeling for managers, *Simulation*, 56(5), 316-320
- Reilly J.P., 1995, Does RAD Live Upto the Hype – Point, *IEEE Software*, 12(5), 24 -26
- Robrets C.A. and Dessouky Y.M., 1998, An overview of object-oriented simulation, *Simulation*, 70(6), 359-368
- Robinson S., 1994, *Successful simulation: practical approach to simulation*, McGraw-Hill, Berkshire
- Rohrer, M.W., 2000, Automod Tutorial, *Proceedings of 2000 winter simulation conference*, Eds. Jones J.A., Barton R.R., Kang K. and Fishwick P.A., Orlando, Florida, 241-245
- Sadowski D.A. and Grabau M.R., 1999, Tips for successful practice of simulation, *Proceedings of 1999 winter simulation conference* Eds. Farington P.A., Nembhard H.B., Sturrock D.T. and Evans G.W., Phoenix, Arizona, 61-66
- Sakthivel S. and Agarwal R., 1992, Knowledge-based model construction for simulating information systems, *Simulation*, 59(4), 223-236
- Sargent R.G., 2000, Verification, validation, and accreditation of simulation models, *Proceedings of 2000 winter simulation conference*, Eds. Jones J.A., Barton R.R., Kang K. and Fishwick P.A., Orlando, Florida, 50-59
- Saunders M., Lewis P. and Thornhill A., 2003, *Research Methods for Business Students*, Pearson Education, Harlow
- Seppanen M.S., 2000, Developing industrial strength simulation models using Visual Basic for Applications, *Proceedings of 2000 winter simulation conference*, Eds. Jones J.A., Barton R.R., Kang K. and Fishwick P.A., Orlando, Florida, 77-82
- Shannon R.E., 1998, Introduction to the Art and Science of Simulation, *Proceedings of 1998 Winter Simulation Conference* Eds. Medeiros D.J., Watson E.F., Carson J.S. and Manivannan M.S., Washington, DC, United States, 7-14
- Sheehan K. and Hoy M., 1999, Using e-mail to survey internet users in the United States: methodology and assessment, *Journal of Computer Mediated Communication*, 4(3)

References

- Shull F., Carver J. and Travassos G.H., 2001, An empirical methodology for introducing software processes, *ACM SIGSOFT Software Engineering Notes* , 26(5), 288-296
- Silva L., Ramos A.L. and Vilarinho P.M., 2000, Using simulation for manufacturing process reengineering – practical case study, *Proceedings of 2000 Winter Simulation Conference*, Eds. Jones J.A., R. R. Barton R.R., Kang K. and Fishwick P.A., Orlando, Florida, 1322-1328
- Smith J.S. and Peter B.A., 1996, Short term scheduling using discrete event simulation, *A paper submitted to Southwestern Bell – Technology Resources, Inc*
- So B. and Lew A., 1999, Automatic code generation for simulating information systems, *Simulation*, 73(3), 159-167
- Sommerville I., 2001, *Software engineering*, Pearson Education Limited, Essex, England
- Son Y.J., Jones A.T. and Wysk R.A., 2000, Automatic generation of simulation models from neutral libraries: An example, *Proceedings of 2000 Winter Simulation Conference* Eds. Jones J.A., Barton R.R., Kang K. and Fishwick P.A., Orlando, Florida, 241-245
- Standridge C. R., 1999, Modular simulation environments : An object based architecture, *Proceedings of 1999 Winter Simulation Conference*, Eds. Farington P. A., Nembhard H.B., Sturrock D.T. and Evans G.W., Phoenix, Arizona, 598-602
- Stanford M. and Graham R., 1998, Are business managers and non-technical consultants ready for low cost discrete-event Simulation? A survey of users, *Proceedings of 1998 winter simulation conference* Eds. Medeiros D.J., Watson E.F., Carson J.S. and Manivannan M.S., Washington DC, United States, 1333-1338
- Subramanian G.H. and Zarnich G.E., 1996, An examination of some software development effort and productivity determinants in ICASE tool projects, *Journal of Management Information Systems*, 12(4), 143-160
- Swain J., 2001, Simulation software survey, *OR/MS Today*, February
- Takus, D.A. and Prfozich D. M., 1997, ARENA software tutorial, *Proceedings of 1997 winter simulation conference* Eds. Andradóttir S., Healy K.J., Withers D. H. and Nelson B.L., Atlanta, Georgia, 541-544
- Tichy W.E., 1998, Should Computer Scientists Experiment More?, *IEEE Computer*, 31(5), 32-40
- Trybula, W., 1994, Building simulation models without data, *Proceedings of 1994 IEEE International Conference on Systems, Man, and Cybernetics. Humans, Information and Technology*, Volume 1, 209-14

References

- Tudhope D., Beynon-Davies P., Mackay H. and Slack R., 2001, Time and representational devices in Rapid Application development, *Interacting with computers*, 2001, 13(4), 447-466
- Tversky B., Morrison J.B. and Bertrancourt M., 2002, Animation : can it facilitate, *International Journal of Human-Computer Studies*, 57(4), 247-262
- Tye B.S., 1999, *A study on model design in the simulation of manufacturing systems*, An unpublished PhD thesis, Sheffield Hallam University, Sheffield.
- Umeda, S., and Jones A., 1997, Simulation in Japan:State-of-the-art update, *Technical Report NISTIR 6040*, National Institute of Standards and Technology, U.S. Department of Commerce, Technology Administration.
- Verner J.M. and Cepra N. ,1997, Prototyping : Does your view of its Advantages depend on your job?, *Journal of Systems and Software*, 36(1), 3 - 16
- Warner T., 1996, *Communication skills for information systems*, Pearson education, Essex
- Weidenmann T., 1999, Database oriented Modeling with simulation microfunctions, *Proceedings of 1999 winter simulation conference* Eds. Farington P.A., Nembhard H.B., Sturrock D.T. and Evans G.W., Phoenix, Arizona, 586-590
- Whitten J.L., Bentley L.D. and Dittman K.C., 2001, *Systems analysis and design methods*, McGraw-Hill Irwin, New York, Fifth Edition
- Xia F., 1998, What's wrong with software engineering research methodology, *ACM SIGSOFT Software engineering notes*, 23(1), 62-65
- Yin R.K., 2003, *Case Study Research*, Sage Publications, Thousand Oaks, Calif., Third Edition.
- Zelkowitz M.V. and Wallace D.R., 1998, Experimental models for validating technology, *IEEE Computer*, 31(5), 23-31.

Appendix I

Questionnaire on Simulation Model Development

Questionnaire on Simulation Model Development

1. What is your **current profession** (Please specify)

2. Do you have a **degree/professional qualification** in the following

<input type="checkbox"/> Production Manufacturing Engineering	<input type="checkbox"/> Mechanical Engineering
<input type="checkbox"/> Operations Research	<input type="checkbox"/> Systems Modelling
<input type="checkbox"/> Software Engineering	<input type="checkbox"/> Business /Management

3. **How long** have you been **building simulation models**

<input type="checkbox"/> Under 5 years	<input type="checkbox"/> 5- 10 years
<input type="checkbox"/> 10-15 years	<input type="checkbox"/> 15-20 years
<input type="checkbox"/> Over 20 years	

4. Roughly, **how many simulation models** have you built in this time? (Please specify)

5. Please **rank** the following simulation project phases in **order of difficulty** (Most Difficult – 1 Least Difficult –5)
 - Problem/Objective definition
 - Conceptual model building
 - Simulation/computer model building and testing
 - Experimentation
 - Project Completion an Implementation

6. Please indicate (with an X) the typical **minimum and maximum percentage of project time** you spend on the following.

	10%	20%	30%	40%	50%	60%	70%
Problem/Objective definition							
Model building and testing							
Experimentation							
Project completion and implementation							

7. Which of the following software do you use to develop simulation models. (Please indicate one or more)

< Special purpose simulation languages

- GPSS SIMSCRIPT SLAM SIMAN
 Any other (Please specify)

< Simulators

- Arena Extend AutoMod Promodel
 Witness QUEST Simul8 Awe Sim
 Any other(Please specify).....

< General Purpose Languages

- Visual Basic C++ FoxPro
 Any other (Please specify).....

8. Do you use any of the following software in Simulation Model Development/Experimentation

- MS_Excel MS_Access Visio VBA

9. Based on your experience indicate your agreement with following statements. (Please mark an X)

	Strongly Agree			Strongly Disagree		
Models are frequently simplified due to limited time allocated for projects						
Sometimes I had to reduce time for experimentation because I had spent more time on model building						
Previously built models are frequently re-used to build new models						
There are large number of people who have not studied a scientific discipline, building simulation models						
New issues arose during the experimentation stage lead me to re-build new models/modify built model						
I build a conceptual model before building a simulation model						

Many Thanks

Please return the questionnaire to

Saman.Wickramaratne-S-Yapa@student.shu.ac.uk

Dear Sir/Madam,

I am in the process of developing a new methodology for accelerating the simulation model development process. I propose to accelerate the Simulation Model development Process by improving the understanding between the user and the modeller. In achieving that I propose to apply Rapid Application Development Techniques in simulation model development. There are three components in the proposed methodology.

1. A tool to develop a conceptual model independent of the simulation software
2. A tool to translate the conceptual model into the simulation model, i.e. into a commercial package such as Arena or ProModel
3. Application of Joint Application Development Team Approach in Simulation Model development

By using the proposed approach it is expected to reduce the simulation model development time by translating the developed conceptual model directly into the simulation model. In this context I like to know the views of the simulation practitioners and academics. I am grateful to you if you provide your views on the above proposed methodology, particularly giving attention to the following questions.

- i. Does the lack of understanding/miscommunication between the user (domain expert) and the modeller (simulation expert) lead to delays in the simulation model development process?
- ii. Do you think the proposed approach will have an impact on the duration of the simulation model development process?
- iii. Practical difficulties you may see in implementing the proposed approach
- iv. Any other observations

Thank you in advance for your time

Please reply to

Saman.Wickramaratne-S-Yapa@student.shu.ac.uk

Saman Yapa,
Researcher
School of Engineering,
Sheffield Hallam University,
United Kingdom
+44-(0)114-2253395

Yapa S. T. W. S., Clegg D.R. and Perera T.D.S., 2004, Conceptual Model: A Methodology for Accelerating Simulation Model Development Process, *Proceedings of the Second International Conference on Manufacturing Research*, Sheffield, United Kingdom

Yapa S. T. W. S., Clegg D.R. and Perera T.D.S, 2005, A New Methodology for Accelerating Simulation Model Development process, *Proceedings of the 14th International Conference on Applied Simulation and Modelling*, Benalmádena, Spain