Loughborough University

This item is held in Loughborough University's Institutional Repository (https://dspace.lboro.ac.uk/) and was harvested from the British Library's EThOS service (http://www.ethos.bl.uk/). It is made available under the following Creative Commons Licence conditions.

creative commons

COMMONS DEED

**Attribution-NonCommercial-NoDerivs 2.5**

**You are free:**

- to copy, distribute, display, and perform the work

**Under the following conditions:**

**Attribution**. You must attribute the work in the manner specified by the author or licensor.

**Noncommercial**. You may not use this work for commercial purposes.

**No Derivative Works**. You may not alter, transform, or build upon this work.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

**Your fair use and other rights are in no way affected by the above.**

This is a human-readable summary of the Legal Code (the full license).

Disclaimer

For the full text of this licence, please go to:
http://creativecommons.org/licenses/by-nc-nd/2.5/

# HYBRID MODELLING METHODLOGY FOR SYSTEM DESIGN

By

Bing Yu

A Doctoral Thesis
Submitted in partial fulfilment of the requirements
for the award of
Doctor of Philosophy
of the Loughborough University of Technology

May 1995

Department of Manufacturing Engineering

© Copyright by Bing Yu 1995

BEST COPY

AVAILABLE

Variable print quality

PAGE NUMBERS CUT OFF IN ORIGINAL

***To My Parents and My Sister's Family***
***For Their Love, Support and Encouragement***

## Acknowledgement

I would like to express my sincere gratitude to my research supervisor, Dr. K. Popplewell, for his guidance, encouragement and interest in this work. I would also like to give my thanks to Professor R. Bell for his support and help. Without this support, I would not stay here to finish my research work.

I would like to thank Mrs. J. Harding for her help. Appreciation is extended to Mr. T. Deave for his useful discussion and great help in my case study.

Finally, I wish to express my love and appreciation to my family and friends for their support, encouragement and friendship throughout the period of my study.

## Synopsis

In the face of rapid development in information technology coupled with a growing dynamism in global markets, manufacturing systems have to be re-constructed for short term or long term goal. Such innovations promise to lead to a new competitive stage, which typically involve design of function, information and behaviour of systems. In order to design the system, simulation has often been chosen. However, simulation has proved limited and fails to aid design of such a complex systems because of consuming much computing time and cost, especially when modelling larger systems. Thus, there is a need to seek a new approach, in a way that results in simulating such a large manufacturing system with less demand on computing time and cost.

This study researches into a hybrid modelling approach to minimise these limitations. It includes proposing a hybrid modelling methodology and developing a hybrid modelling tool. The methodology integrates simulation and metamodelling techniques. The metamodel employed in the study possesses, not only characteristics of conventional metamodels in terms of representing relationships in quantity, but also in time lapse. This is the originality of the study and the significant distinction between this research and application of metamodelling in conventional ways. The hybrid modelling tool is developed to support and demonstrate the identified hybrid methodology. LISP has been used as the software language for the hybrid modelling tool. The result of this work concludes that the hybrid modelling approach is capable of simulating a complex manufacturing system with less demands on the computer.

The work reported in this thesis has been carried out in conjunction with the EPSRC research project, Hierarchical Manufacturing System Modelling (HMSM) (GR/F96549), to produce an Integrated Design and Modelling Methodology (IDEM). The project was initially a collaborative research program including Loughborough University of Technology (LUT), Morris Crane Ltd., of Loughborough and GEC Large Machine, of Rugby. The experience of these collaborators has proved most valuable in supporting the research, and have provided a cross section of views and comments. The research reported in this thesis is set in the context of the HMSM Research group at Loughborough.

# Contents

***Declaration***

***Acknowledgement***

***Synopsis***

***Contents***

**Chapter 1 Introduction** 1

**Chapter 2 Scope of the Research**

2.1 Introduction 6

2.2 Description of the Problem 6

2.3 Research Objectives 9

2.3.1 Research Approach 10

2.3.2 Case Study 11

**Chapter 3 Literature Review**

3.1 Introduction 13

3.2 Factory Design 13

3.2.1 Concept of Factory 14

3.2.2 Dynamic Performance and Measurements 16

3.3 System Design Methodologies 17

3.4 Discrete Event Simulation 27

3.4.1 Simulation Language and Simulator 28

3.4.2 Knowledge Based Simulation 30

3.4.3 Simulation Validation 31

3.5 Metamodelling 33

3.5.1 Validation of Metamodel 35

3.6 Time Series Approach 37

3.7 Evaluation of Statistical Software 38

**Chapter 4 Background**

4.1 Introduction 41

4.2 The HMSM Project 42

4.2.1 IDEM methodology 42

4.2.2 IDEM tool 47

4.3 Relationship Between the Project and the Research 52

**Chapter 5 Hybrid Methodology**

5.1 Introduction 54

5.2 Overview of Major Tasks 54

5.3 Simulation Model 58

5.3.1 Mechanism in IDEM Simulation 61

5.3.2 Function of An IDEM Box vs. Function of A Detailed Modelling Page 63

5.4 Embedding a Metamodel in Simulation 66

5.4.1 Requirement of Embedding Metamodel in Simulation Environment 67

5.4.2 Requirement of Embedding Metamodel in IDEM Simulation Environment 68

5.5 Metamodel 70

5.5.1 General Regression Metamodel 70

5.5.2 Metamodel Validation 74

5.5.2.1 Metamodel Validation Method 74

5.5.2.2 Statistical Test 74

5.5.3 Characteristics of the Metamodel 77

**Chapter 6 Implementation**

6.1 Introduction 80

6.2 Aims of Implementation Work 81

6.3 Implementation Environment 82

6.3.1 Experimental Software Development Environment 82
6.3.2 Metamodel Development Environment 83
6.3.2.1 Zoom Node Library 84
6.3.2.2 Zoom Node Specialisation 87
6.4 Implementation of Hybrid Methodology 90
6.4.1 Implementation of Metamodel 90
6.4.1.1 The Features of the Data Collector 90
6.4.1.2 Data Collector Implementation 92
6.4.1.3 The Features of the Metamodel Generator 93
6.4.1.4 Implementation of Metamodel Generator 93
6.4.1.5 Metamodel Validation 94
6.4.2 Implementation of Hybrid Model 96
6.4.2.1 The Features of the Metamodel Box 96
6.4.2.2 Implementation of Metamodel Box 102
6.4.3 Interface Customization 106
6.5 Experimental Application of Embedding Metamodel 109
6.5.1 Design for One Input and One Output Simulation Scenario in Zoom-In Paradigm 110
6.5.2 Design for Multiple Inputs and One Output Simulation Scenario in Zoom-In Paradigm 113
6.5.3 Design for Multiple Input and Multiple Outputs Simulation Scenario in Zoom-In Paradigm 117

**Chapter 7 Case Study**

7.1 Introduction 120
7.2 The Company Description 120
7.2.1 Operation of the Company 121
7.2.2 Products and Production Procedures 122
7.3 The Case Study 124
7.3.1 Problem Tackled 124
7.3.2 The Description of Production Function 125
7.4 Experiment Approach 125

7.4.1 Simulation Modelling 126

7.4.2 Metamodelling 131

7.5 The Results of Case Study 132

7.5.1 Simulation Model Representation 132

7.5.2 Development of Metamodel 141

7.5.3 Validation of the Simulation Model and the Metamodel 142

7.5.4 Hybrid Model vs. Simulation Model 144

7.6 Case Study Experiences 145

7.7 Conclusion 146

**Chapter 8 Evaluation and Discussion**

8.1 Introduction 148

8.2 Achievements of the Research 148

8.3 Limitation of the Research 149

8.4 Discussion 151

**Chapter 9 Conclusion**

9.1 Introduction 154

9.2 Original Contribution 154

9.3 Summary of the Research Results 154

9.4 Further work 155

9.5 Conclusion 156

**Reference**

**Appendix**

**I Object Oriented Technique**

**II Metamodelling Program**

# Chapter 1

# INTRODUCTION

To keep pace with global competition, manufacturers endeavour to increase their factory competitive capability. This involves not only to improve the end products but also to rethink the organisational structure which is used to manufacture the products. The organizational structure is influenced by both technologies and manufacturing strategies applied. When new manufacturing strategies are investigated from time to time, there is a need to explore organizational structure of manufacturing systems in both static and dynamic states (Cook, 1991).

The complexity of the manufacturing systems arises from many directions: the inter-dependence of elements of the systems, the influences of external forces and the lack of predictability to the consequences of actions. Because of the complexity of the manufacturing systems, it is difficult to identify the critical influence affecting the strategic direction. Many aspects have to be considered and tested before a strategic direction can be taken. Obviously, it is not realistic to test all possibility of new proposals in actual system because of cost and time invovled. In order to do so, a model is used. Traditionally, operational research methods are adopted to construct a model. However, it is very difficult and sometimes even impossible to build a model with the methods when the object is a complex system like a manufacturing system. Because a) getting a reliable and accurate model requires large volume of reliable data, b) it is hard to calculate when many variables and interaction are involved. Hence,

computer simulation is considered a fast and cost-effective alternative which is used to test various strategic directions of a manufacturing system to satisfy the objectives of corporations.

Current simulation packages, however, are restricted at either lower level modelling, such as at cell or station level, or a description of the performance of the whole factory in a static way rather than a dynamic one. Hence, there is a strong need for a multi-level modelling technique to be developed. It should provide fast and approximate modelling to discover what is necessary in order to modify the current performance of the whole factory and to achieve the desired manufacturing strategy. Due to this requirement the project, Hierarchical Manufacturing System Modelling (HMSM), was proposed.

The HMSM project offers a new multi-level modelling method. It also provides an associated tool -Integrated DEsign and Modelling (IDEM)-by which a factory model can be built approximately and extended to investigate detailed operations. Like any other simulation package, because IDEM generates results by performing many iterations of a set of variables, this package requires enormous computer power. Though computer technology has been improved dramatically both in memory and speed, it is still a mammoth task to model a system such as a manufacturing one. Thus, it is necessary to add metamodelling techniques to enhance the capability of IDEM tool.

The main subject of this thesis is, therefore, to develop a tool which integrates

simulation and metamodelling techniques to be able to model complicated manufacturing systems.

The thesis is arranged into nine chapters including this introduction. Its structure is shown in Figure 1.1. An overview of each chapter is listed as follows:

Chapter 2 states the scope of the research. It identifies the research objective. In order to help the user to understand this research objective, the problem is described briefly. Finally, a description of the research approach is discussed. In order to understand the objectives better, this chapter is related to chapters 3 and 4 which provide the literature and technical background.

Chapter 3 begins with an extensive literature review of areas related to this research work. These are system design and analysis methods, discrete event simulation, and metamodelling. These methodologies are also related to practical applications. In addition, the time serial method is discussed.

Chapter 4 gives the reader the background to the research of the HMSM project. It contains the objectives of the project, the methodology and the IDEM tool developed in the project. The user should then be able to understand the simulation environment where the metamodel is going to be developed and where metamodel replacement takes place.

Chapter 5 explains the theory of the research. The major tasks are reviewed to refresh

and emphasise the domain of the research. This is followed by analysing the principle of the simulation. This leads to reveal the philosophy of embedding a metamodel in a simulation. Then, the requirements are identified and illustrated, which is essential for embedding metamodel in a simulation environment. Finally, the methods which have been employed for creating and validating a metamodel are examined theoretically.

Chapter 6 is devoted to the implementation of the hybrid methodology. To understand the development of the experimental software, the implementation environment is depicted first. Thereafter the implementation is illustrated through generating metamodel, the interacting metamodel and replacing the simulation model. It contains rather simple examples of implementation in order to illustrate the experimental software implementation clearly.

Chapter 7 is a case study. In the beginning of this chapter, the industrial company is introduced briefly. Its simulation model and hybrid model are then described. The results of the study are illustrated at the end of the chapter.

Chapter 8 summarises the research work. The achievements of the study are addressed, and limitations of the work are also discussed.

Finally, key issues of the research work are concluded in chapter 9. Recommendations for possible further research are also presented.

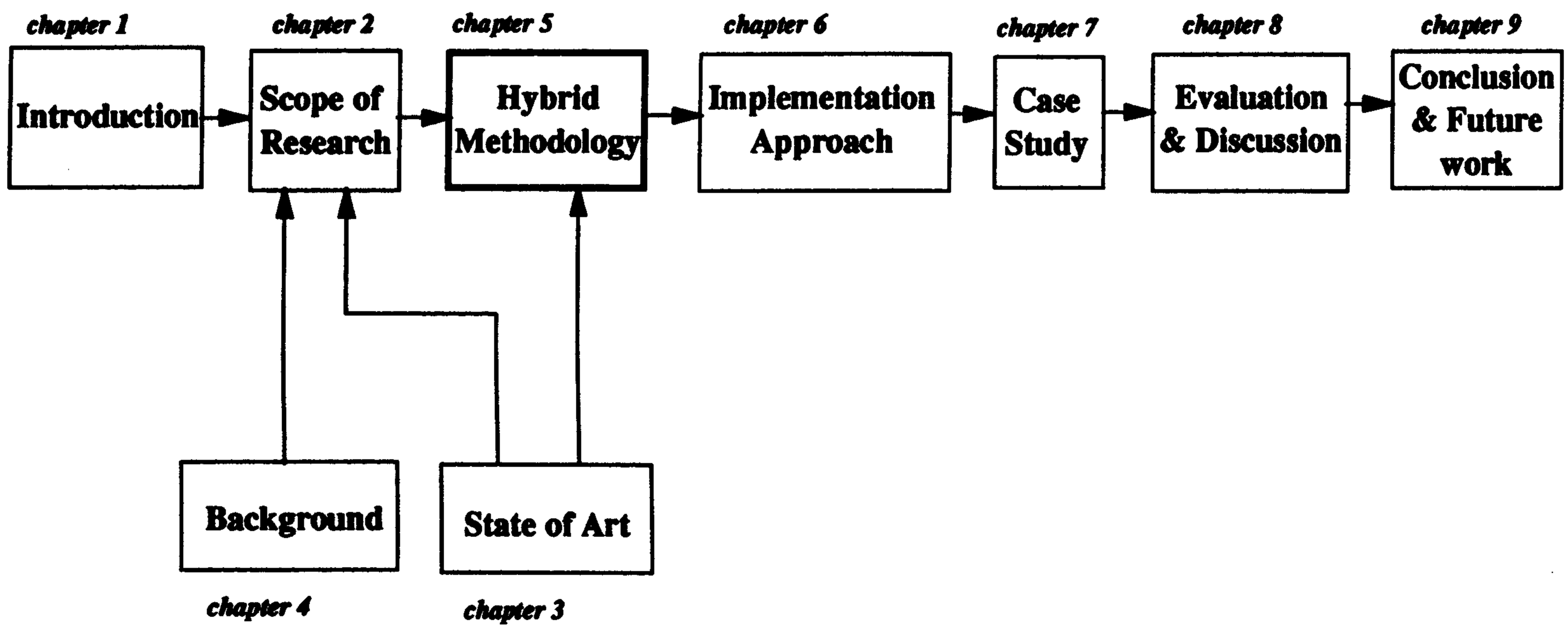



**Figure 1.1 Structure of the Thesis**

# Chapter 2

# SCOPE OF RESEARCH

## 2.1 Introduction

This chapter presents the objectives of the research on the integration of both metamodelling and simulation techniques in the modelling of factory system design. Initially, it outlines the problems. Then, the objectives and the research approach are illustrated respectively.

## 2.2 Description of the Problem

To achieve a competitive edge in the world-wide marketplace, many companies have to adjust both their business objectives and the organisation structure so that the business strategy can be executed and accomplished. In the new manufacturing environment, the long term management objectives are profitability, increased manufacturing flexibility, product innovation and lead time reduction. In pursuing the goal, many companies adopt advanced techniques and a new control mode such as Just In Time (JIT) and Total Quality Control (TQC). Although some of them are successful in improving partial performance of a manufacturing system, the majority fail to improve their market position as expected (Hanson, 1992). Therefore the problem of how to design and analyse an overall system has arisen in the research field. Evidently, there is a strong need for a factory design tool which can investigate business strategy

and manufacturing strategy mutually, and reveal the system behaviour in its totality.

In order to investigate the impact of different strategies operated in an existing or proposed system, simulation which implies discrete event simulation has been used in favour to conventional methods. Many reports (Zulch and Grobel, 1992) (Maxwell et.al., 1983) have shown that it is a powerful tool for the planning, design and control of complex systems. Because it is a set of well-designed experiments, simulation requires tremendous computer power and time. Therefore, most simulation models are limited to a small area such as a cell or in the modelling of the processes of a work station. Although results of analysing partial systems can assist us in gaining knowledge, it is fragmented. It cannot provide a global picture of the system where an attempt is made to modify current performance so as to achieve the desired business goal, to analyse the success of actions to accomplish change, and to evaluate merits and demerits under different proposed strategies. However, if the system studied contains too many variables, current computer power is not capable of providing an adequate analysis within the available time and / or resources (Shanon, R. E. 1992). How to simulate a large system to assist managers in getting insight into their objectives is becoming increasingly important and critical.

As the knowledge of system behaviour can be expressed in quantity, application of the metamodelling method has been re-considered recently (Yu and Popplewell, 1994). A metamodel which uses a mathematical formula to generalise simulation results instead of running a simulation model is recommended (Freedman, 1987) (Madu, 1990). Because it is a mathematical model which takes much less computer memory and time

to execute, it can overcome the disadvantages of simulation. Hence, it can offer an opportunity for modelling a large system. In addition, because simulation is a trial and error method, it does not provide a method of optimisation. To put it simply, it results in indirect guidance as to ways of optimising system performance. In contrast, the metamodelling technique is able to indicate the impact of each independent variable on a response variable by revealing the relationship between the dependent and independent variables of the system studied.

However, metamodelling depends on a large volume of statistical data, in the absence of which it would be either difficult to represent the relationships of a system, or even impossible to identify the relationships. Because simulation is a trial and error method, a system behaviour can be knowledged via a set of experiments. In other words, the lack of need for a large volume of statistical data is an advantage of the simulation. It is very useful in a case where various strategies or operations cannot be implemented in the actual system by reason of time and cost. Therefore, the author of this dissertation argues that there is a huge potential value of integrating metamodelling and simulation techniques in order to increase the possibility of modelling a larger system.

Certainly, metamodelling is not an entirely new technique. In the literature, it has always been applied in post-analysis of simulation results in order to interpret the simulation, or to gain the knowledge to guide further simulation experiments. But it has not been found that simulation and metamodelling techniques can be put into operation together to treat system design. Although many researchers gain knowledge of their simulated system through metamodelling which can be generated by an existing

statistical package, such as SAS, Minitab etc., it is not easy to put both simulation and metamodelling into practice in a single environment unless the user either writes his/her own program to transfer simulation data to construct a valid metamodel, or input simulation data tediously from a keyboard. Obviously, there is a lack of tools which assist users in employing both simulation and metamodelling techniques consistently.

As a consequence, it is necessary to establish a hybrid methodology which allows us to model a large system. To demonstrate the methodology, a supporting tool which integrates the capabilities of simulation and metamodelling is needed. These will be the topics for discussion in the later chapters.

## 2.3 Research Objectives

**From the above brief overview of the problems, it is clear that the main objective of the study is to research into embedding a metamodel into a simulation environment, to enable us to design and analyse a complicated manufacturing system by means of simulation.**

In order to achieve this aim, following are the objectives of this research work:

- to apply a metamodel in a simulation environment to take the advantages of metamodelling to overcome imperfections of using existing statistical packages.
- to insert a metamodel in the simulation to produce a hybrid methodology

which enables the simulation of larger systems in less time and with reduced cost.

- to develop a tool to explore the feasibility of supporting the hybrid methodology.
- to demonstrate an efficient and economic methodology by which knowledge is gained for decision support.

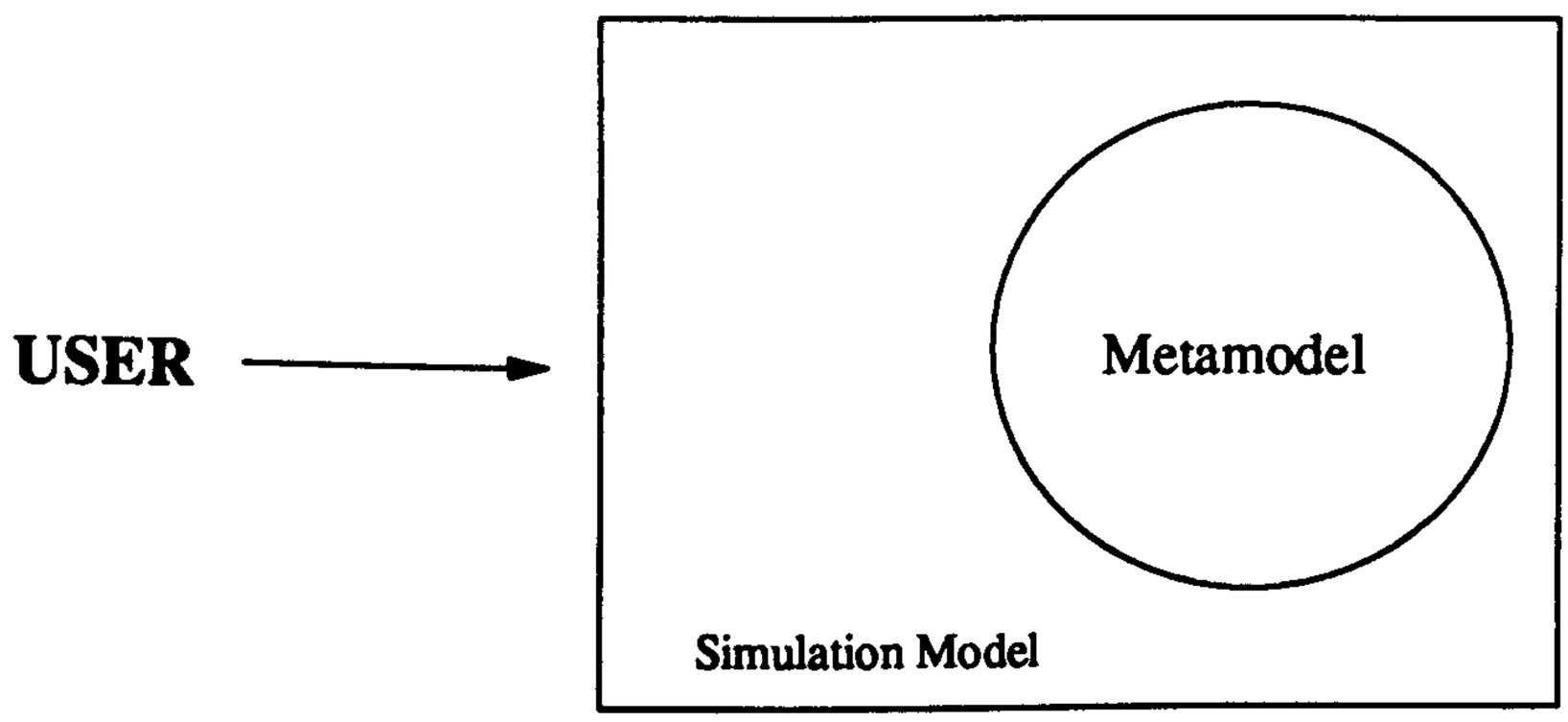



**Figure 2.1 Embedded Metamodel in Simulation**

## 2.3.1 Research Approach

In designing a manufacturing system, a new hybrid methodology is first proposed. It should possesses the following features:

- the capability of the object-oriented programming and technique of statistical analysis

statistical analysis

- approximate system description
- low cost and efficient evaluation

There is also a need for a tool to allow the user to do it within IDEM software. The features of the tool are composed of the following:

- menu driven for various options
- handling input and output simulation results easily
- icon indication
- flexible replacement between simulation and metamodel

To fulfil the tasks, the research takes three stages, these are (a) building a simulation model of a manufacturing system (b) generating a metamodel according to a detailed simulation model where knowledge of the behaviour of the system is needed, (c) replacing the simulation model with the corresponding valid metamodel.

Figure 2.2 shows an overview of this research.

### 2.3.2 Case Study

To test the methodology in practice, an industrial case is studied. Since a problem is beyond the scope of this dissertation, a dynamic problem of the shop floor has been chosen for study.

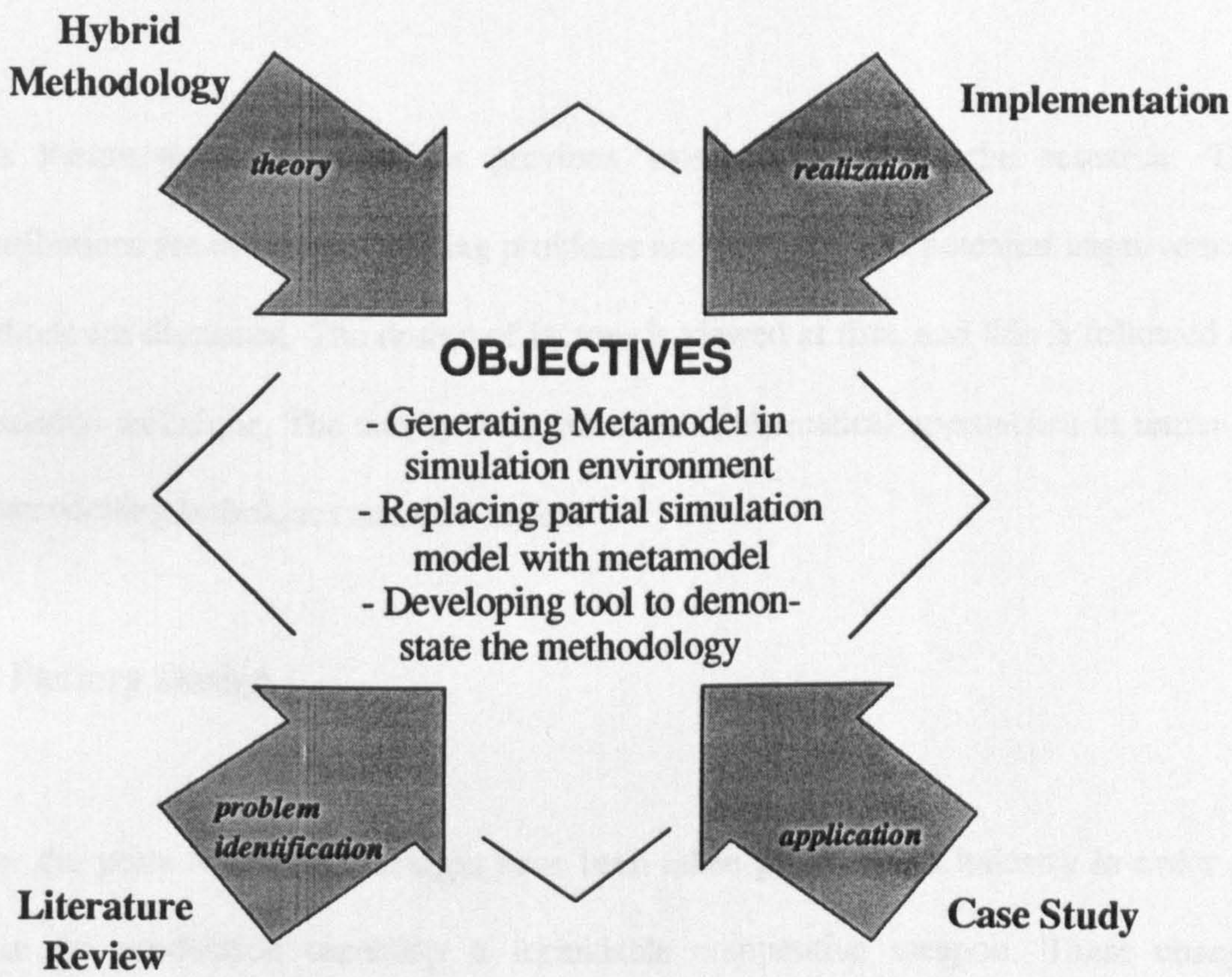



**Figure 2.2 Overall of Research Objectives**

# Chapter 3

# LITERATURE REVIEW

## 3.1 Introduction

This literature survey examines previous works relevant to the research. The contributions are evaluated, existing problems are identified and potential improvement methods are discussed. The design of factory is viewed at first, and this is followed by simulation technique. The survey then examines mathematical approaches in terms of metamodelling techniques and time series.

## 3.2 Factory Design

Over the years the radical changes have been taken place within industry in order to make the production capability a formidable competitive weapon. These changes encompass a entire range of activities that begin with market demand and end with customer satisfaction (Chaharbaghi 1990). Those activities are not only within the factory but also extend to external collaborators, for example vendors. Therefore, there is no doubt that a factory design methodology is needed for re-design of existing manufacturing systems, or developing entirely new plant to ensure a manufacturing system remaining competitive in the short and long term.

In this section, infrastructure and facilities of a factory have been examined at first. The

dynamic behaviour of a factory has then been viewed in different dimensions. The issues related to this research work are discussed. Finally it considers current design methodologies.

### 3.2.1 The Concept of the Factory

There are two types of manufacturing systems: conventional manufacturing system and computer integrated manufacturing system (CIM). However, whether a conventional manufacturing system or a computer integrated manufacturing system (CIM) is applied, every factory comprises the products its produced and the facilities which are used to manufacture the products. Every factory is committed to delivery customer satisfaction as its ultimate goal. The differences between the conventional factory and CIM lie in the use of information system and the organisational by the information system and the application of advanced technologies. Because a manufacturing system can classified differently by using different methods, the definition of CIM will also vary accordingly.

In general, a factory's internal organisation can be classified into sales, marketing, engineering and manufacturing. Each of these sectors is independent in function but works as a totality. The functions and the interactions between them are represented in **Figure 3.1**.

In different cases, the definition of them changes. Prutte *et al.* (1988) categorises the facility into four: marketing, accounting, production and inventory. Rembold *et al.* (1989) classify into marketing research, long-range planning and forecasting, capital

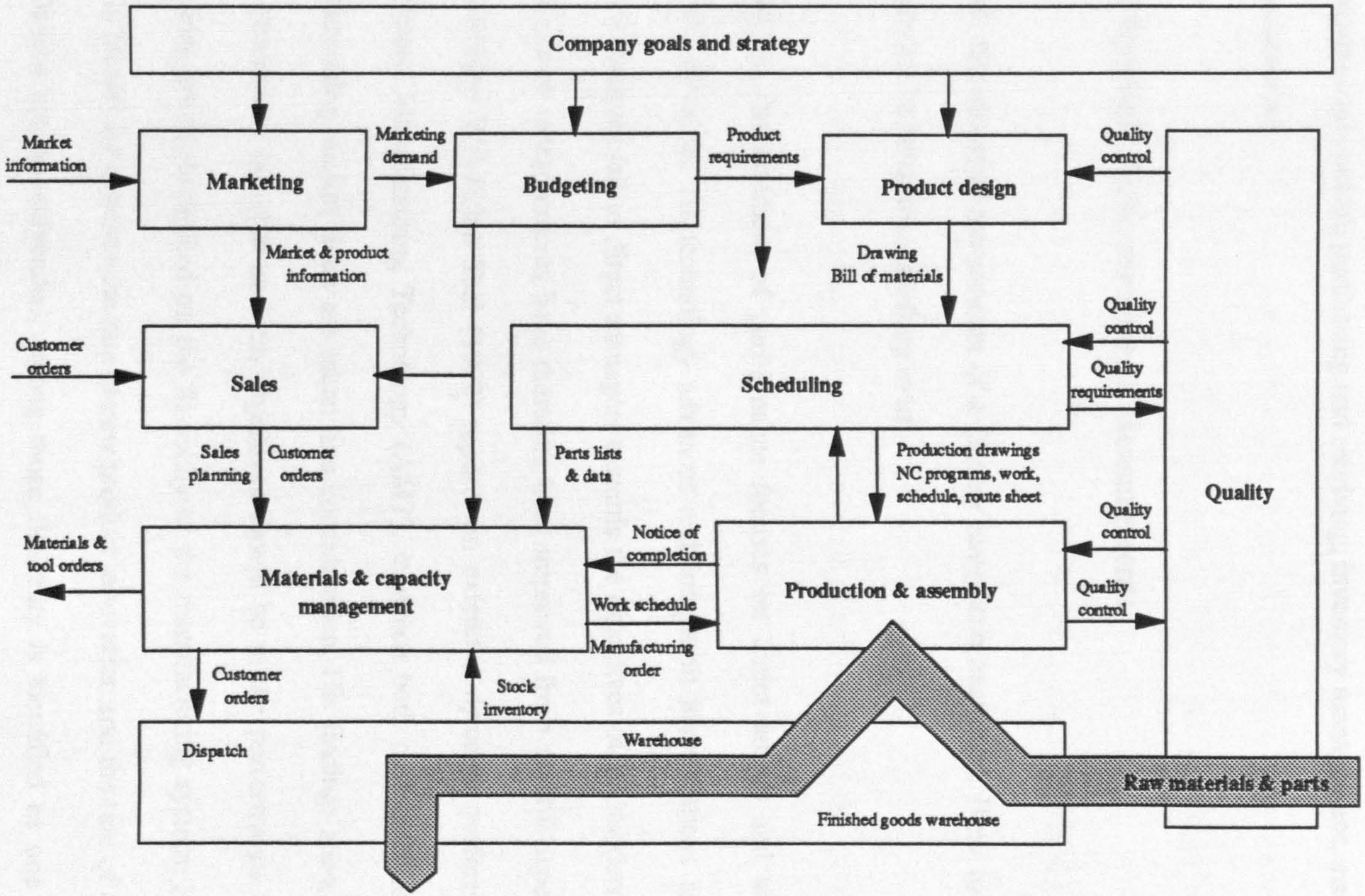



**Figure 3.1 Major Functions in a Manufacturing Factory (adopted from Rembold, 1991)**

equipment and facility planning, customer order servicing, engineering and design, manufacturing process planning, production order planning and manufacturing monitoring and control, purchasing and receiving, inventory management, maintenance and accounting.

### 3.2.2 Dynamic Performance and Measurements

So far, the essential components of a factory have been examined. How to measure and predict its behaviour is left up to us.

Traditional measurement of performance focuses on direct labour and short-term financial elements. As technology advances and long-term investigations are made, these techniques fail to direct managers towards the objectives of the modern factory. Performance measurements have therefore been improved from several aspects, such as flexibility. Wei-Hsien Tsai (1992) reports on extended dynamic performance on Automated Manufacturing Technology (AMT), in which both customer satisfaction and increasing market share are taken into consideration. His findings show that not only production but also business objectives should be in the performance measure. Paraphias group researched on the flexibility of the manufacturing system. From this certain factors for instance the rate of new product innovation and the rate of customer satisfaction are recommended, among those flexibility is identified as one of most important features in factory design.

Focusing on estimating business value from the re-structure of the organisation

measures the effects of investment in re-engineering on the different aspects of enterprise performance, such as profitability, flexibility and competitiveness. The primary economic impact of re-structuring an organisation occur at the operational levels, such as cost reduction and flexibility, these effects may then be traced though the value chain to affects at the enterprise performance level, such as profitability.

Different areas vary with certain parameters for which state or behaviour can be revealed. These can be quantities or qualities.

Performance measurement is important on system design. If a correct performance measure is chosen, the system can be redesigned or modified into an adequate and reliable model, from which alternative strategies can be evaluated.

Lead time, flexibility and so on are always counted in any type of factory. Machine break down and other factors which have critical impact on production are also caught.

## 3. 3 System Design Methodologies

Since concepts of a manufacturing system and performance of a manufacturing system have been discussed, it is now necessary to identify the techniques which can be used to design a manufacturing system in a systematic way.

CIM-OSA (Open System Architecture for Computer Integrated Manufacturing). CIM-OSA is a well-known CIM architecture. It uses enterprise models for the

monitoring and control of daily enterprise operation. For this purpose CIM-OSA is developed in two major parts: an executable enterprise model and an integrated infrastructure. The total enterprise model is viewed in respect of its function, information, resource and organisation, see **Figure 3.2.** The model should be generated in a mode of enterprise derivation which is supported by enterprise requirements definition, enterprise design specification and enterprise implementation description. The enterprise implementation description is the model which will be released to the integrating infrastructure for execution of the daily enterprise operation. Three levels of generalisation will be supported by generic building blocks, partial model and particular model. (Jorysz and Vernadat, 1990) (Klittich, 1990).

Though the method is often used to produce a reference model, only function and information views are completed and presented at moment. The behaviour and time representations are insufficient (Wortmann, 1994).

**GRAI methodology** (Graph a Results et Activities Interlies).

The development of GRAI is based on the argument that using formal rules in the early phases. It will reduce errors in design and therefore improve overall system performance. It has three phases: analysis phase, design specification phase and operation-level phase.(Doumeingts et al., 1987)

This approach, however, does not encourage the designer to check for technological, economical or financial feasibility during the design process. Also there is no reference to company strategy, or the multitude of other 'business' questions which should be

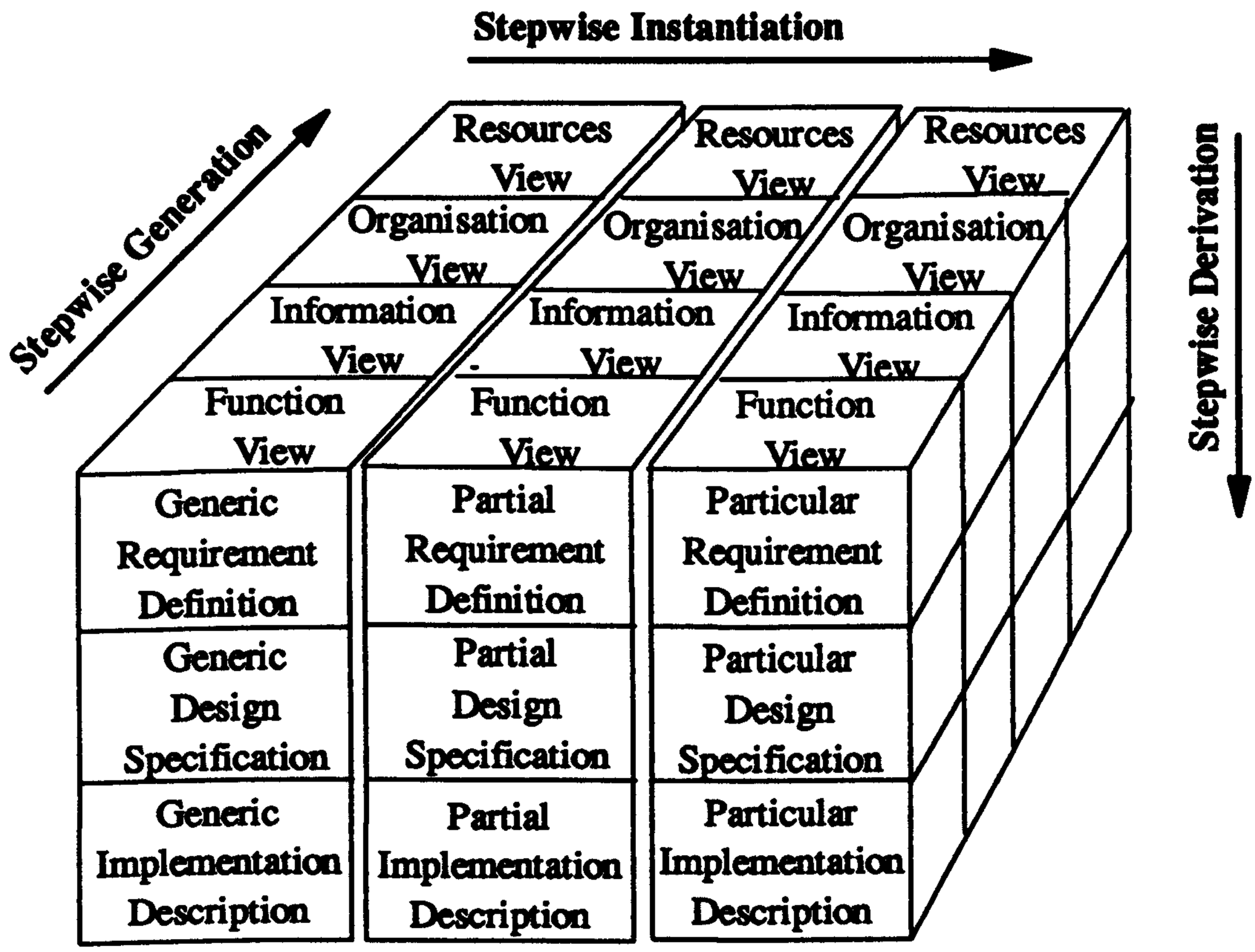



**Figure 3.2 Overview of CIM-OSA Architecture Framework (adopt from Jorize, 1990)**

PAGE

MISSING

IN

ORIGINAL

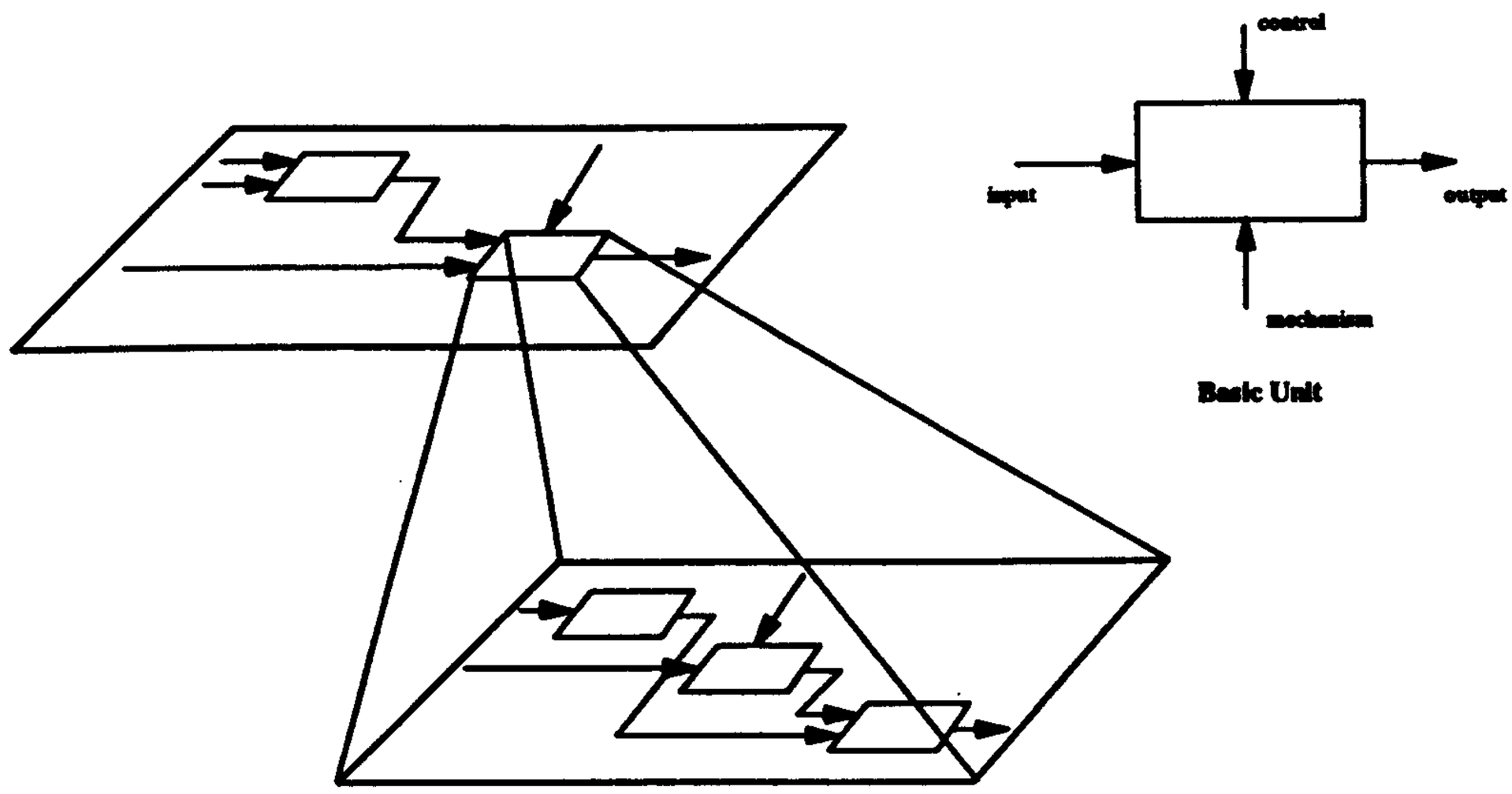



**Figure 3.5 IDEF0 Methodology**

function (or activity) and arrows denote flow of data between activities. Input data to the left of the box is transformed and results are output from the right of the box. Control arrows to the top of the box denotes the constrains that affect performance of the activity. Mechanism arrows to the bottom of the box represent the resources that carry out the activities. To overcome the lack of information modelling, $IDEF_1$ has been designed. It is developed for design of integrated database. However, its lack of an inherent means of describing inter-functional entities (Bravoco and Yadav, 1985b) led to the development of $IDEF_{1X}$. $IDEF_{1X}$ is a graphical semantic information modelling technique which defines the logical structure of shared data in terms of entities, attributes of entities and relationships between entities (SofTech Inc., 1979) (Mackulak, 1983) (Hicks et al., 1986). The limitation of $IDEF_{1X}$ on dynamic modelling initiates the creation of $IDEF_2$. $IDEF_2$ is focused on improving the productivity of simulation modellers(Bravoco and Yadav, 1985c)(Sarkis and Li, 1994).

In summary, these methodologies adopt a top-down approach. Some of them have been considered and adapted widely. The opposite of a top-down approach, is bottom-up design. By convention, bottom-up approach implies putting all the information and data which exists at low levels into higher level. The higher level model can then be testified and justified. Although this way has been adopted during the system design process, it is more like a feedback or design life cycle than a methodology. In fact, there is no evidence that a methodology for bottom-up approach yet exists.

A comparison of each system design methods is illustrated in **Table 3.1**.

Meyer *et al.* (1987) concluded that GRAI-net formalism has proved to be a valid tool for knowledge acquisition relating to hierarchical planning, sub-goal interaction, and constraint inter-dependence. However, this approach must be oriented according to the specification, which is not transferable.

In the report given by Hicks *et al.*, IDEF methodologies have been used for modelling a CAPM system. It is argued that the performance of CAPM systems critically depends on the "model" of reality, stored as data within the system, being realistic. The IDEF methodologies are helpful at the design stage, but operational factors also have to be taken into account.

Colquhoun et al. (1989) described how $IDEF_0$ can be used to link design and manufacture in a CIM environment. This linkage can be established by use of computer aided process planning. The $IDEF_0$ model has defined the information flow necessary

| | *CIM-OSA* | *GRAI-GIM* | *IDEF* | *SADT* | *SSADM* |
|---|---|---|---|---|---|
| **Function** | Yes | without behaviour | Yes (IDEF0) | Yes, with activation rules | partially (not behaviour) |
| **Data** | Yes | Yes | Yes(IDEF1x) | Yes, partially | Yes |
| **Decision** | Limited | Yes | Some representation possible | Yes | No |
| **Time Attribute** | Limited | Yes | Yes(IDEF2) | No | No |
| **Simulation** | No | No* | Yes(IDEF2) | No | No |
| **Graphic Representation** | Yes | Yes | Yes | Yes | Yes |

**Table 3.1 Comparison of System Design Methodologies**

* It is subject of future development.

manufacture in a CIM environment. This linkage can be established by use of computer aided process planning. The $IDEF_0$ model has defined the information flow necessary to support each activity so that functional and information flow requirements can be established for the computer system at each level defined in the ISO proposed model. The IDEF technique is also recognised to aid the implementation of changes (Bravoca and Yadav, 1985a) (Baines and Colquhoun, 1990).

Modelling complex system interactions in a manufacturing enterprise is a current research topic. In practise, a graphical model of the order-to-ship (OTS) process was built using hierarchical process modelling (HPM), derived from $IDEF_0$. The model is focused on the flow of information, material and control, and results in "the OTS model yielded a way to promise of the potential of showing the impact of complex interactions among different parts of the manufacturing enterprise". (Mujtaba, 1994).

Except for the above well-known methodologies, a schematic modelling concept has been introduced to offer an advanced simulation methodology for system design. The appeal of this schematic modelling concept is its simplicity and flexibility. The methodology gives the user the versatility of having as many conditions as are necessary to define in any desired way, i.e. interactions of functions performed by objects within a system. (Chaharbaghi, 1991)

It is often useful to study a dynamic real-world system to learn something about its behaviour. However, it is generally necessary to use a model to study the performance of the system first, since experimentation with the manufacturing system itself would be either disruptive, not cost-effective or simply impossible. An appropriate model

could shed some light on this question by allowing the operation of the plant to be studied as it currently exists and as it would be if the plant were expanded. A model which is composed of entities and the relationship between the entities can be constructed using mathematical methods, if the relationship is simple enough. Otherwise simulation can be used if the relationships are complex.

## 3.4 Discrete Event Simulation

In aid of system design, certain tools are available. Simulation is one of them.

Simulation, refer to Shannon (1984), is the process of designing a model of a real system and conducting experiments with this model for the purpose either of understanding the behaviour of the system or evaluating various strategies for the operation of the system. Therefore, a simulation model is an abstracted representation of real system which attempts to indicate problems and solutions to the problem by using a trial and error approach.

According to the time changing patterns, simulation models can be classified into following three types:

- discrete event simulation
- continuous simulation
- combined simulation

Since a manufacturing system performance can be considered, in a modelling environment, as continuous processes and discrete events, simulation here refers

A simulation model may be built using either a general-purpose language (e.g., FORTRAN, BASIC) or a simulation language, which can be further divided into general-purpose simulator and specified model.

The advantage of using a language - like FORTRAN is that it offers the user the ultimate flexibility to model virtually any system; also the language is probably already known by the user and probably available on the user's computer. On the other hand, simulation model written in a general-purpose language can take a tremendous time to develop since the language is not particularly oriented toward simulation modelling. In contrast, using a simulation language accelerates a model building process because of certain modules or features offered.

### 3.4.1 Simulation Languages and Simulators

Simulation languages that are applicable to manufacturing problem may be further classified into two categories, general-purpose simulation languages and manufacturing-oriented languages / simulators.

General-purpose simulation languages, such as GPSS (H, V, or PC), WITNESS/SEE WHY, SIMAN/CINEMA, SIMSCRIPT II.5, SLAM, are useful for simulating wide variety of systems (e.g. health system and motor way service system) in addition to manufacturing systems, but may contain certain features specifically for manufacturing. These languages, which exhibit discrete event simulation features, allow an analyst to develop a simulation model of a manufacturing system in less time than would generally be required when using a language like FORTRAN.

Using GPSS/H, Felix (1992) constructed a simulation model to evaluate the performance of a welding assembly line which requires modification to achieve some of the philosophies of a JIT manufacturing environment.

Balancing and design of an assembly line was also researched by Rajamani (1991). A model, constructed with SLAM II, was used to determine the in-process storage capacity for physical layout. The use of the simulation model conveys to management the relationship between in-process inventory and average time a product spends in the system. Other typical examples are SIMAN (Norman and Scheck, 1986) (Banks and Carson II, 1985), SIMSCRIPT (Russell, 1986), GPSS (Beerel, 1987) (Schriber, 1986), and SLAMII (Khoshnevis and Chen, 1986) etc.

Certain simulation software packages, called simulators, are specialised. The simulators which have been designed for simulating manufacturing-type problems are of interest here. They include SIMFACTORY II.5, MAP/1, ProModelPC and AutoMod. SIMFACTORY II.5 is a factory simulator written by SIMSCRIPT II.5 and designed for engineers who are not full time simulation analysts. It can simulate either layout or all processes that are involved from workstation to transportation (Jiao, 1993). MAP/1, similar to SIMFACTORY II.5, is a actually a simulator rather than a language since a particular system within an available class of manufacturing systems are modelled by entering data rather than doing programming. ProModelPC, likely being situated between simulator and language, has more programming features within the environment, and has the capability to add C or Pascal type subroutines to a program. Manufacturing features such as AGVs and conveyors are captured as well. Comparison with the others, AutoMod is very powerful in its description of material

handling functions.

Other simulation languages or simulators that are designed for particular purpose such as one-of-a kind production system design can also be found. A simulator viz. FEMOS is an example. Market competition has forced manufacturers to change their production strategy from producer-oriented strategy to customer-oriented strategy. Since this type of organisational concept and technical measures are enmeshed on the one hand with major changes in production systems and the other with possibly high financial risk, the probable impacts must be estimated thoroughly before implementation.

### 3.4.2 Knowledge Based Simulation

Knowledge based simulation focuses on building a knowledge base and its management by using an inference engine. The knowledge base and inference engine are provided as distinct and independent entities permitting the modeller to alter the components independently(Layery, 1986).

In the simulation modelling, some of the AI techniques, such as rule-based reasoning, frame-based representation and object-oriented programming, have been widely used to drive the simulation (Langen, 1987). Rule-based reasoning uses "IF-THEN" scheme so the programming is easily understood. The frame-based representation allows the incorporation of details of each object and its relationship to other objects in the system more easily. Any object under the frame-based representation scheme could have any number of procedures which can be activated by various means. This means

any object knows its behaviour. This is the concept of object-oriented programming.

Ben-Arieh (1986a) (1986b) has developed a knowledge based simulation system for exploration into the routing of jobs in a multi-cell FMS. Palaniswami (1992) studied and demonstrated a particular application by integrating a knowledge system with simulation. It focused on making scheduling decisions in a manufacturing environment. The knowledge system was constructed using the M.1 expert system package and the simulation was performed using SLAM II. Other examples can be found (Lim et al., 1987) (Doumeingts, Darricau and Berard et al., 1987) (Wang, 1989).

Currently, overwhelming interest in object-oriented approach to simulation modelling has been recognised. There are many object-oriented methodologies available in the literature, among them Booch (1989) and Rumbaugh (1987) are popular. Champeaux et al. (1988) compared twelve object-oriented analysis and design methodologies and described their differences. Monarchi et al. (1989) evaluated 23 object-oriented analysis and design methodologies and identified their strengths and weaknesses. Korson et al. (1990) proposed a unifying paradigm for object-orientation.

### 3.4.3 Simulation Validation

One of the most important issues in simulation studies is the model validation process, which determines whether a simulation model is an accurate representation of the actual system or not. Unfortunately it has not been studied thoroughly in the past (Cochran, 1988)(Law and Kelton, 1982).

One of the validation methods is a multi-step approach which has been recommended by several authors as a general guideline (Cochran 1988) (Law and Kelton 1982)(Naylor and Finger 1967). The primary objective of the first step is to develop a model which can produce reasonable results to those people who are knowledgeable about the system under studied. Existing theory should be used for the modelling, such as the determination of probability distributions of customer inter arrivals, which has been proven to be the Poisson process (Law and Kelton, 1982). The second step is to use statistical procedures to test quantitatively the assumptions made during the first stage of modelling. The third step tests the ability of the model to predict the system under study. Two alternatives available for the procedure are historical validation and validation by forecasting (Cochran, 1988). This is the last step used to determine how representative the simulation output data are to the real system. However, it is often difficult to compare simulation output data with those from the corresponding real system. First reason is that the real historical data may or may not exist. Second one is that they may be difficult or even impossible to collect. Besides statistical procedures, people who are knowledgeable about the system can be asked to examine the simulation output data. If the people can differentiate them with real system data, their explanation can be helpful in improving the model validation.

A different approach of validation is to construct a confidence interval for the output data (Law and Kelton 1982). If the real value of the variable, obtained either from actual system observation or from established theory is within the confidence interval, the simulation model can be considered reasonable.

## 3.5 Metamodelling

In general the simulation modelling method is ideally suited to reveal details of a complex system where interactions and relationships between inputs and outputs can hardly be identified at primary design stage. However, it requires a great amount of time and cost in computer power and the results do not give any optimised solution. Moreover, an industrial manager, who may not be a simulation expert, would have difficulties in understanding simulation results.

Because the principle of simulation is a trial and error approach, several problems arise from it. The first problem is that it is time consuming (Law, 1989). This is particularly obvious if several replications or long runs are needed to compare numerous scenarios (Palaniswami, 1992). Secondly, simulation is memory consuming, particularly when running a complex and hierarchical system model (Palaniswami, 1992). Thirdly, simulation results often need interpretation or statistical treatment in order to be utilised. Because numerous or large tables of results do not provide the user with understandable relationships between the simulation model inputs and its responses.

Since there are many limitations of simulation modelling, metamodelling has been proposed to enhance simulation capability. It can also be seen as a mathematical approach. According to Keijnen (1984), metamodelling implies an auxiliary model constructed with the regression analysis method. The detailed principle of metamodel will be discussed in chapter 5. Because it can establish relationship between inputs and outputs of a system in quantity, it has attracted more attention recently (Yu and Popplewell, 1994).

Blanning's (1984) model for direct sensitivity analysis is perhaps one of the earliest examples of metamodelling. Here an inventory model is discussed, and three metamodels relating to cost, service level and buffer stock ratio are investigated.

Application to an FMS has been demonstrated by Keijnen where the FMS used four different machines to carry out three types of operations to produce a product. The results of this metamodel can indicate where the system bottleneck is. This experiment of regression metamodel has also provided an insight into the work of FMS ( Kleijnen, 1988).

There are a range of reports from manufacturing system (Rajamani and Singh, 1991) (Goyal 1991) (Patrick 1991) to green house system (Rotmas and Vrieze, 1990) in the literature. Each of them exhibited the use of a metamodel for simulation post-analysis and proved that it is a worthful post-analysis tool to help to understand the simulation model. Rajamani and Singh (1991) concluded that " the metamodel helps the manager obtain an idea of the time a product spends in the system at different failure rates and the number of repair personnel required after testing". Patrick's work describes and endorses the use of such an approach to find good parameter values which become the initial setting to fine-tune the simulation and thereby accelerate experimentation.

A maintenance float problem has also been studied by Christan Madu et. al. (1992). The metamodel is formalised with factorial experimental design. As given combination of repair person and standby units, the model can predict the average equipment utilisation of a maintenance float system.

Eilon and Mathewson (1973) also emphasise the need for more aggregated models,

besides detailed simulation models and propose to use network models instead of regression metamodels.

Mar and Bakken (1981) and Geoffrion (1976) emphases the need for metamodelling (outside simulation) in a mathematical programming and system dynamics, respectively. Hendry (1983) presents metamodels for the generalization of statistical studies in econometrics.

Jothishankar group (1992) explored the use of philosophy of JIT in real manufacturing system with metamodelling. The results of their research indicate that "metamodels provide useful tools to operations managers for identifying the combination of different variables to achieve minimal throughput time."

Examples of metamodels used to analyse simulation models can be found in Meisel and Collins (1973), Ignall et al. (1978), Goldstein and Dushane (1976), Sharbird (1990), and Friedman and Friedman (1985).

Despite many applications of metamodel have been found, most of them use metamodel for post-analysis, some of them solely use metamodel to simulate a system. No case has been found in which both simulation and metamodel technique works in a single environment.

### 3.5.1 Validation of Metamodel

Once a simulation model has been developed, the need for validation is still very

important.

One validation method is to develop a regression model using only two thirds of the observations random selected (Friedman, 1985). The remaining third, treated as the holdout part, is used to test the regression model. Both selected and unselected cases is tested by evaluating the $R^2$ (coefficient of determination). If the two $R^2$s are very close, it implies that the model has a good predictive validity and can be expected to perform well on real data.

Double cross-validation is a technique which is a variation on the above method (Friedman, 1985). In this method, data is randomly split into two parts of approximately equal size, so called "split-halves". Then, a comparison is made on the two $R^2$ values. If they are very different, it means that the metamodel is probably not valid. On the other hand, similar $R^2$ indicates that the metamodel developed on the full set of data may be used.

The application of cross-validation has been applied for the selection of regression model can be seen among Atkinson (1982), Cook and Weisberg (1982), Hocking (1983), and Hocking and Pendleton (1984).

Snee (1977) suggests a different split of the observations into estimation data (used to obtain ) and prediction; that is, he does not split the n observations into n-1 estimation points and 1 prediction point, and he does not apply permutation.

For cross-validation, also see Bunk and Drog (1984), Friedman and Friedman (1985),

and Narula and Wellington (1985).

## 3.6 Time Series Approach

Simulation and regression analysis methods are appropriate to serve a wide range of steady state applications. Real-time systems have features that distinguish them from other computerised systems (Ledbetter and Cox, 1985), hence they require real-time modelling methods and methodologies. To accommodate their real-time nature, a number of extensions to general purpose modelling methods (such as a data flow based methodology) have been proposed. The extensions, developed by Ward and Mellor (1985) and Hatley and Pirbhai (1986), enable the analyst to represent control flow and control processing as well as data flow and processing. It also includes state transition diagrams for the processes. In addition, time series is suitable for real-time systems design since it deals with the analysis of a sequence of data, usually in time order. From a technology point of view, Priestley (1981) defines time series as a stochastic process with finite second moments. The feature of time series analysis that distinguishes it from other statistical analysis is the explicit recognition of importance of the order in which the observations are made. Many of the time series development have been contributed from the field of statistic and mathematics, although they have widespread ranging from engineering to social science.

Priestley (1981) and Jenkins and Watts (1968) worked in the theoretical frequency domain. In contrast, this is the time domain which is based on Wold (1980) decomposition theory and can be found among Newbold (1981) (1984). Two papers describing later research relating to frequency and time domain respectively could be

obtained from Brillinger and Kishnaiah (1983) and Hannan et al. (1985).

MRP under the uncertainty demand and the change of lot size rules has been investigated with time domain approached by Ip (1993). It reveals that demand variation and lot size rules have a significant impact on MRP performance. The relationship can be adequately represented using the time series methods. The results of it can be taken as knowledge for manufacturing system design. Finally, he concluded that "integrated simulation and time series approach appears to be a powerful methodology for the design and analysis of complex manufacturing systems."

Both Buzacott (1985) and Cheng (1985) have researched on real-time control of assembly line balancing and resource allocations by using simulation. However, application of these results are of limited value since they do not capture the vital time information about the system. To meet the time demands in real-time control, Cochran and Li (1993) have developed a decision support system in which the time series method is applied to express and predict dynamic transient behaviour of an assembly line so as to assist a manager making decision. They reported that "the resultant empirical representations reveal a surprising accuracy."

## 3.7 Evaluation of Statistical Software

As already mentioned, a metamodel is a type of regression model so that a software which has functions to generate regression model can be used. Here three popular software package, SPSS, SAS and MiniTab, are evaluated. The results of comparison of the three packages is displayed in Table 3.2.

**SPSS** (Statistical Package for Social Science) is a general statistical analysis package, a detailed description of the SPSS can be seen in (SPSS-X user guide 1986). The regression modelling is available in the Regression Procedure of SPSS. The procedure consists of two steps, Identification and Estimation. The user is required to identify, by checking the statistic data such as coefficients and standard error, whether or not the regression model is valid (SPSS-X, 1986)

**SAS** (Statistic and Analysis Software) is another statistical software which has the advanced forecasting module in Econometric and Time Series Analysis. SAS includes the ARIMA procedure, STATE SPACE procedure and spectral analysis for modelling univariate and multivariate time series (SAS, 1985)

**MINITAB** is an interactive statistics package written and distributed by Minitab Inc., Pennsylvania, USA. The package offers a variety of statistical methods and can be used for data manipulation and selection by various conditions. Regression Analysis is one of the functions offered. Regression Analysis procedure for metamodelling is simple. It only involves data entry, regression analysis and validation. The validation can be performed by checking the residuals. The regression method includes Stepwise with analysis of residuals (Ryan, et al., 1985).

| | SAS | SPSS | MINITAB |
|---|---|---|---|
| Methodology | Regression analysis and Time series | Regression analysis and Time series | Regression analysis |
| Multivariate transfer function | Yes | No | No |
| Differencing / transformation | Yes | Yes | Yes |
| Graphical data analysis | Acceptable | Acceptable | Acceptable |
| Menu driven | No | No | No |
| Easy to use | No | Acceptable | Acceptable |
| User knowledge | High | Medium | Low |
| Programming capability | Yes | Yes | No |

**Table 3.2 Major Feature of Existing Statistical Package**

# Chapter 4

# BACKGROUND

## 4.1 Introduction

The challenge from international and national markets is more serious than ever. It forces manufacturers to design or re-think their manufacturing systems to enhance their competitive capability. This capability can be described and evaluated in terms of its flexibility, efficiency and reliability. The difficulty for management is how to achieve and maintain a suitable manufacturing system design in both the short and the long-term.

To solve these problems with minimum cost and maximum efficiency, simulation has been used for many years. In order to make this technique user friendly, several methodologies have been reported and a lot of simulation software were established. However, these softwares either succeeds in system description alone or solves only a narrowed dynamic system problem, such as a cell or shop problem in the manufacturing environment. The software systems do not treat the system as a whole.

The HMSM (Hierarchical Manufacturing Systems Modelling) project[1] was carried out at LUT in order to research into these problems and develop an associated tool. The research objective in this thesis is part of the objectives of the project, and the author is

[1] This project is supported by the ACME Directorate of the Science and Engineering Research Council.

one of the members of the team.

## 4.2 The HMSM Project

The HMSM project seeks to address the problem of supporting the design of a factory-scale manufacturing system. The approach adopted is to develop a methodology in which a manufacturing system can be viewed broadly, and can be designed coherently.

The project has three objectives. The primary one is to propose an integrated modelling methodology appropriate for application in the design of new factories, or the redesign of existing factories. The methodology should also facilitate the evaluation of existing manufacturing systems from a factory-wide view of point. A secondary objective is to design a prototype software tool which is capable of supporting the modelling methodology at various levels and demonstrate the feasibility of such a software. The third and last objective is to offer users considerable assistance in the interpretation of results derived from modelling. Such assistance serves both as a guide to the validity of conclusions which may be drawn at any stage, and also directs attention to areas of the model where the degree of approximation or uncertainty should be reduced before valid conclusions can be drawn.

### 4.2.1 IDEM Methodology

The Integrated Design and Modelling (IDEM) methodology is a multi-level multi-view

method. At each different level, certain important aspects of the system are captured, and different but related viewpoints are applied. **Figure 4.1** illustrates the scope of the IDEM methodology. Although it was developed for manufacturing system design, it could be used to analyse or design the structure of other general system. The method has the following major phases:

1. **Conceptual Modelling Phase**. This is a study of the current structure and behaviour of the system and interpretation of new business target. This results in constraints and possible inconsistencies of the current system being identified and the specification of the requirements of a new system in order to meet the business objectives. A boundary is then determined in which design activities can take place. It is clear that the boundary works as a reference model to guide the design of a model in the simulation modelling phase.

2. **Simulation Modelling Phase**. After the reference model has been set up, the aim of this phase is to design a system which possesses the desired characteristics. This is done by using simulation techniques. This phase can be subdivided into two levels:

(a) **Approximate modelling level**. An approximate factory model is developed which focuses on global views of the system, whilst omitting details. An approximate model of a manufacturing system is thus described from three different but related viewpoints: the function view, the information view and the dynamic view. **Figure 4.2** shows the concept of each view point concerned and the correlation of the three view points. The multi-view approach allows a user or modeller to observe the behaviour of the designed system from the different viewpoints. The benefit of the model is that it is

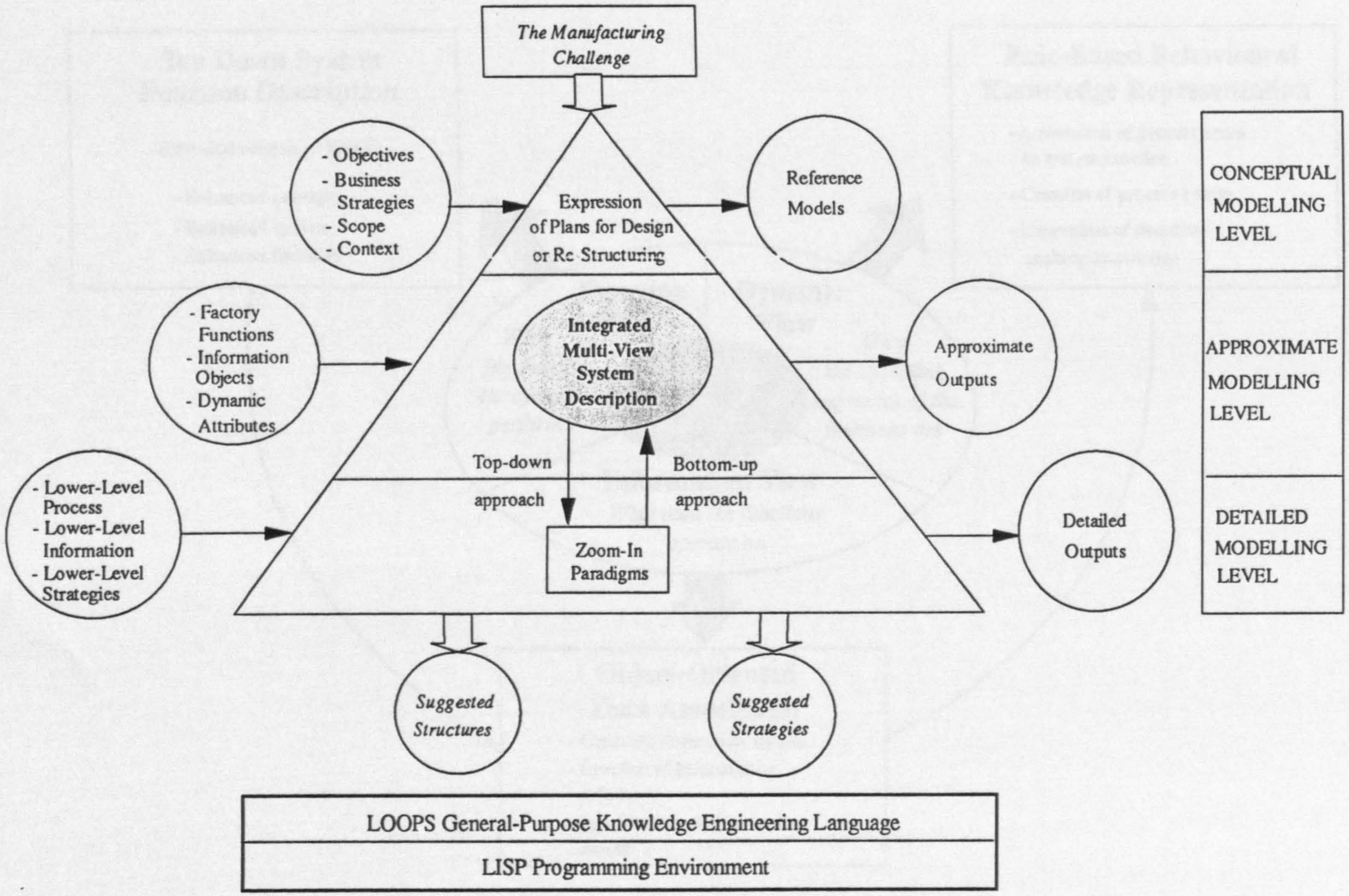



**Figure 4.1 Scope of IDEM Methodology**

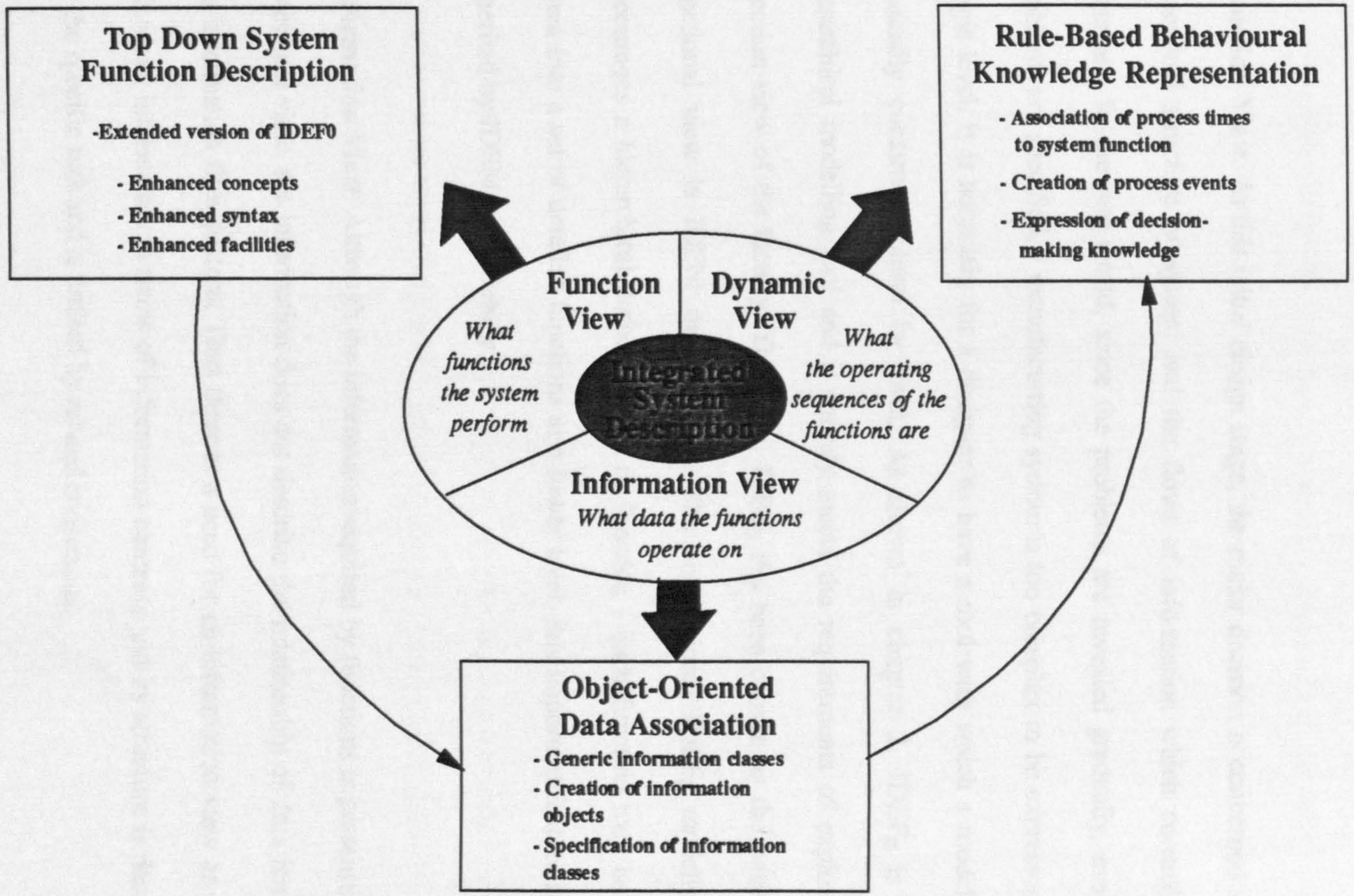



**Figure 4.2 Multi-View System Description Approach**

able to identify sub-systems likely to be critical in the performance of the system and also identifies organisations and activities which need to be investigated further.

**Function View.** At this initial design stage, the major decision is concerned with the functional structure of system and the flows of information which co-ordinate its functions. In the real world, since the problems are revealed gradually, even if the problems are specified, a manufacturing system is too complex to be expressed at one single level. It is necessary for a designer to have a tool with which a model can be gradually constructed level by level. As shown in chapter 2, $IDEF_0$ is such a hierarchical modelling tool and it entirely meets the requirements of exploring this function view of the factory. Therefore, $IDEF_0$ has been chosen as the basis of the functional view in IDEM methodology. The conventional $IDEF_0$ modelling tool encourages a hierarchical decomposition of function - each function may be broken down into a set of detailed functions at a lower level, this important feature has been inherited by IDEM methodology.

**Information View.** Although the information required by functions is presented at the functional view, the information does not describe the relationship of data items. It is the information flow pattern. Thus there is a need for an information view as well. In this view, information in terms of information contents and its structure is determined by the specific task and is limited by related constraints.

**Dynamic View.** The last view of the IDEM approach is concerned with the behaviour of the factory in terms of time attributes. It means each of the lowest level boxes

should be assigned transformation information. This information should contain operation time for each process and should be applied to receive inputs. Through the input of process information into each function box together with the information upon which it acts, a simplified and abstracted existing manufacturing system is completely formed. The performance of a system can be tested and the results of operating such a system can be analysed.

(b) **Detailed Modelling Level**. Based on the approximated model, an extended model is constructed by adding details to selected areas of the approximated model. This can provide the linking mechanism between sub-system simulations, allowing the study of their interactions. Here activities of each entity are the major concern. This level also includes the information view and the dynamic view with more details.

By using a combination of the multi-view and multi-level approach together, a simulation model of a factory can be generated. However, a system design may not be carried out all at once. For instance, after the analysis of the dynamic behaviour of the designed system, new problems may be identified and better understanding of known problems may be gained. This leads the designer to reconsider the established functional structure. To make this design cycle easy, IDEM methodology provides a bottom-up approach which is important in maintaining the model during the design life cycle.

### 4.2.2 IDEM Tool

Having viewed IDEM methodology, a prototype software tool has been developed to fulfil and support the methodology.

**I. Approximate Modelling:**

**Function view.** It is a functional diagram with flows of information associated with these functions in the $IDEF_0$ form. An example is given in **Figure 4.3**. In terms of factory design a box denotes a function not a department, arrows denote flows of information and flows of physical material associated with a function.

Though conventional $IDEF_0$ includes flows of information, it does not constitute an information model. It neither shows the structures within flows nor the relationships between information entities. Thus, to achieve the requirements and to make transferring the function model into an information model possible, three approaches to the development of the IDEM function view have been adopted: conceptual, syntactical and the provision of facilities. In addition, three types of label have been identified and provided.

**Information view.** Information is the data the functions operate on. This view is brought about by means of support facilities for inputting and editing the data related to the established hierarchical functional model. Because of the extension of the IDEM function view, each IDEM label or box of the functional representation may enter data directly and has a data handling function which gives a user the ability to specify

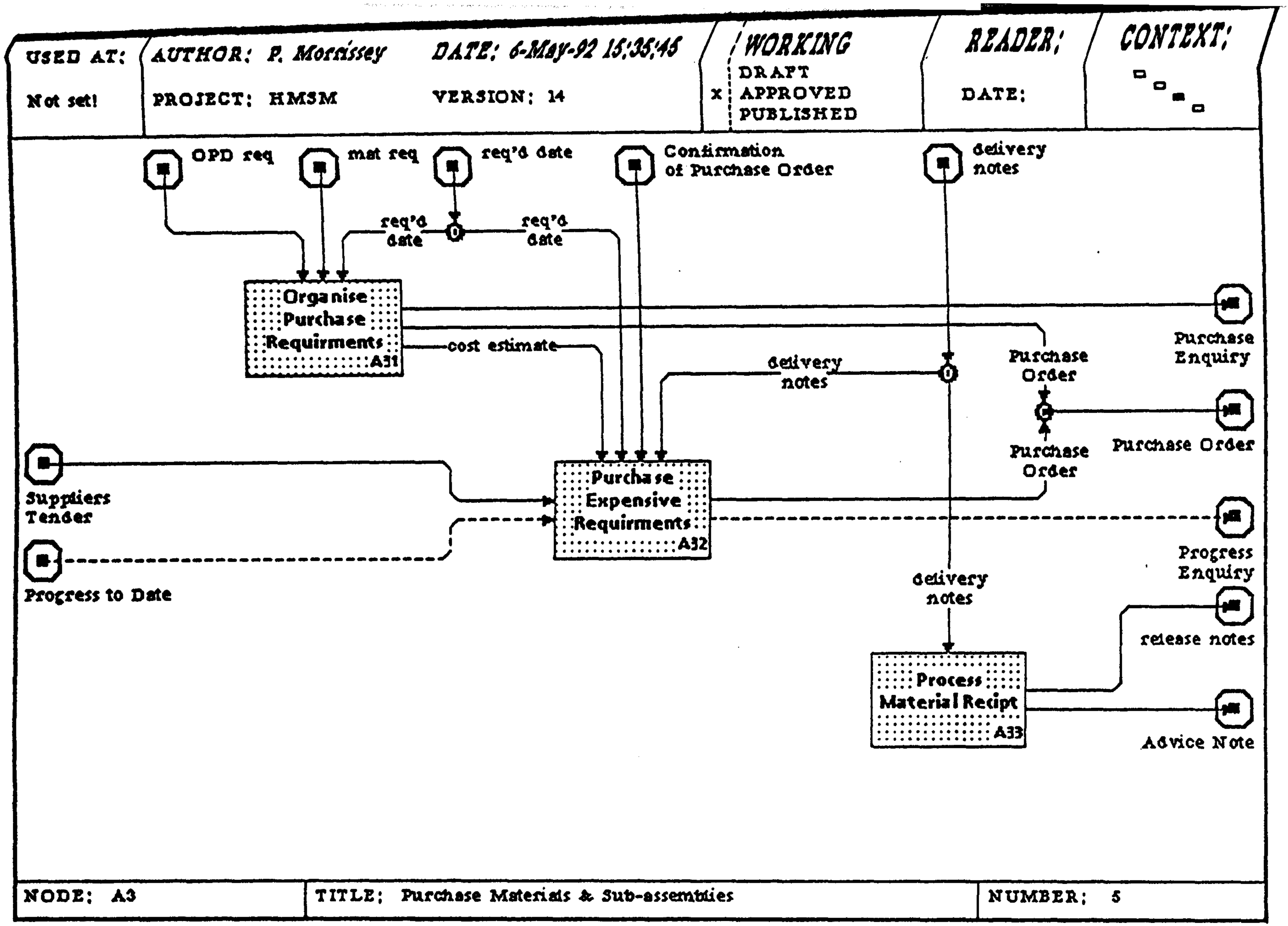



Figure 4.3 An Example of Function View

matching data. The information includes knowledge of the arrival patterns, and the value of items associated with both arriving and despatching information object.

In the lowest level IDEM box, information is classified into input flow information and output flow information. Although both of them have flow name in common, but the latter also contains flow image and flow data, while the former carries only flow input gate. The matching function, the data handling function and the package selecting function are treated as information as well, and can be identified by a unique name.

**Dynamic view**. In this work, this means the provision of facilities by which the user can express behavioural and decision rules which operate on the data entered during the construction of the information view. Each IDEM box has a defined behavioural rule. In addition, the IDEM methodology also provides a facility to allow users to specify their own specialist behavioural rules.

At a high level, a decomposed function does not need behavioural rules but does require route information between a set of substitute functions.

**Performance output**. This is a necessary and important part of the IDEM tool. There are two categories, debugging and performance reporting. It can also be divided into time related and quantity related performance.

Within this modelling stage, the IDEM tool provides a formatted structure for the information and dynamic views. Thus the tool is easy to use for a user who has less

knowledge of modelling and the environment.

## II. Detailed Modelling

In IDEM methodology, detailed modelling level is also called the zoom-in paradigm. Here, a user can develop or create their own sophisticated knowledge and data base without the frame, which is offered in the approximated modelling stage. Although this requires the user to have great knowledge of the LISP and LOOPS, it is a powerful tool which gives the user a chance to do precisely what he would like.

In contrast with organisation structure and information associated with the organisation being the design objective at the approximate modelling level, time attributes and individual processes are the main concerns in this design level. Detailed information and activities, such as how many parts are to be manufactured, what operations to be made and what production rule to be applied, are simulated in order to understand the behaviour of the system. Thus, time scales at this level are smaller, and the requirements for computer processing times and memory could increase dramatically according to the complexity and size of the model.

Since many fields in manufacturing systems have been researched, certain knowledge relating to commonality can be found. The project offers a zoom node library in order to simplify the design process. The library consists of a set of nodes, which represents an object in the real world, each of which possesses essential activities and characteristics of the object.

## 4.3 Relationship Between the Project and This Research

By using the IDEM methodology and software, the organisation and performance of an existing or proposed manufacturing system can be investigated. Because simulation is a comparison technique for various input pattern combinations, validation is a very important stage. To validate a large system model such as a manufacturing system, requires an effective validation tool. A metamodel attempts to reduce the demands on consuming resources, therefore metamodelling has been considered as one of objectives of the IDEM project. Thus, it can be seen that this work has initially been developed based on the IDEM project.

A second link to the IDEM methodology. Figure 4.1 illustrates the IDEM methodology which includes both -- top-down and bottom-up approaches. Since IDEM methodology has been classed as top-down, the exploration of approaches to include a bottom-up approach links the project work and the author's research. In a conventional way, the bottom-up approach implies that user steps back to investigate another part of a model at an higher level based on all detailed information and rules of a sub-model. Although it is often used in real design practice cycle, it is not a methodology rather than a design manner. The bottom-up approach here differs significantly from the conventional one in concept, whilst adopting the same terminology, as this indicates the design direction clearly. To be precise, the bottom-up approach in IDEM is an abstract modelling approach. The approach implies use of a simpler model, which can generate adequate simulation results and takes less computation time. These results can then be substituted into a detailed sophisticated

simulation model. Once a sub-system model has been replaced, no more detailed information or rules of the sub-system are kept in the new model. Therefore, the need for computing time and memory is decreased when the new model is executed. In other words, a more detailed simulation model can be built with the same computer.

Therefore, looking for a simpler model is the key in the research. It is well known that mathematics provides an alternative to constructing a model using a simulation technique. Also the mathematical model consumes less time in computing. Metamodelling is such a technique which has been shown capable of generating one kind of simple mathematical model. Thus, the core of the bottom-up methodology is embedding a metamodel into a simulation.

Finally this research aims to demonstrate the feasibility of the abstract bottom-up approach and of reducing resource consumption. In order to do this, a tool is needed and this should be compatible with the IDEM tool.

# Chapter 5

# HYBRID METHODOLOGY

## 5.1 Introduction

This chapter is going to describe the hybrid methodology proposed in this study. The major tasks are reviewed in the first section. Then, it focuses on the three main principles, namely simulation, embedding a metamodel in simulation environment and metamodelling respectively. In the simulation part, the essential principle relating to the embedded metamodel is reviewed. This is followed by the philosophy of embedding a metamodel in simulation. Finally, the essential concepts of theoretical exposition of metamodel and validation of a metamodel are explained.

## 5.2 Overview of Major Tasks

The internal and external challenges force industries to innovate their businesses and organisations to secure their market position. Many critical factors have to be identified in order to make correct new policies to face the challenge of the real-world. This depends largely on the overall performance of a factory rather than part of it (Love et al., 1992) (Parnaby, 1991).

Simulation has been widely used as a design tool to model a whole process in aid of evaluating and testing new proposals. However, if the model is getting bigger and

bigger, it will impose a great demand on computing power. This weakness limits its capability of modelling a larger system, such as a manufacturing one. On the other hand, because simulation is an experimental technique consisting of a set of trial-error experiments with a variety of inputs combinations, it allows a system analyst to model a system initially without mathematical formula. By studying the results of the experimentation, the analyst can gain a rough and preliminary knowledge of the system.

A metamodel, which use mathematics to explain the behaviour of a system, makes less demand on computer's memory and speed. A metamodel, which is one type of analytical model, can expose characteristics of a system accurately in terms of revealing the effect of each input on the output in quantity. In addition, a metamodel can be employed for getting a quick estimate of the expected simulation response at an input value not yet observed.

It is clear that simulation model can provide an analyst with a chance to describe the system in order to reduce the complexity at the first stage of design and metamodelling techniques will give opportunities to accelerate a simulation trial.

Considering the advantages of both simulation and metamodelling, a hybrid methodology is proposed for a large system design. A prototype tool is then developed to prove its feasibility.

The hybrid methodology attempts to minimise the weakness of the simulation by merging both techniques into a single environment. The framework of this research approach is illustrated in **Figure 5.1(a)** which mainly provides a user with a road map to

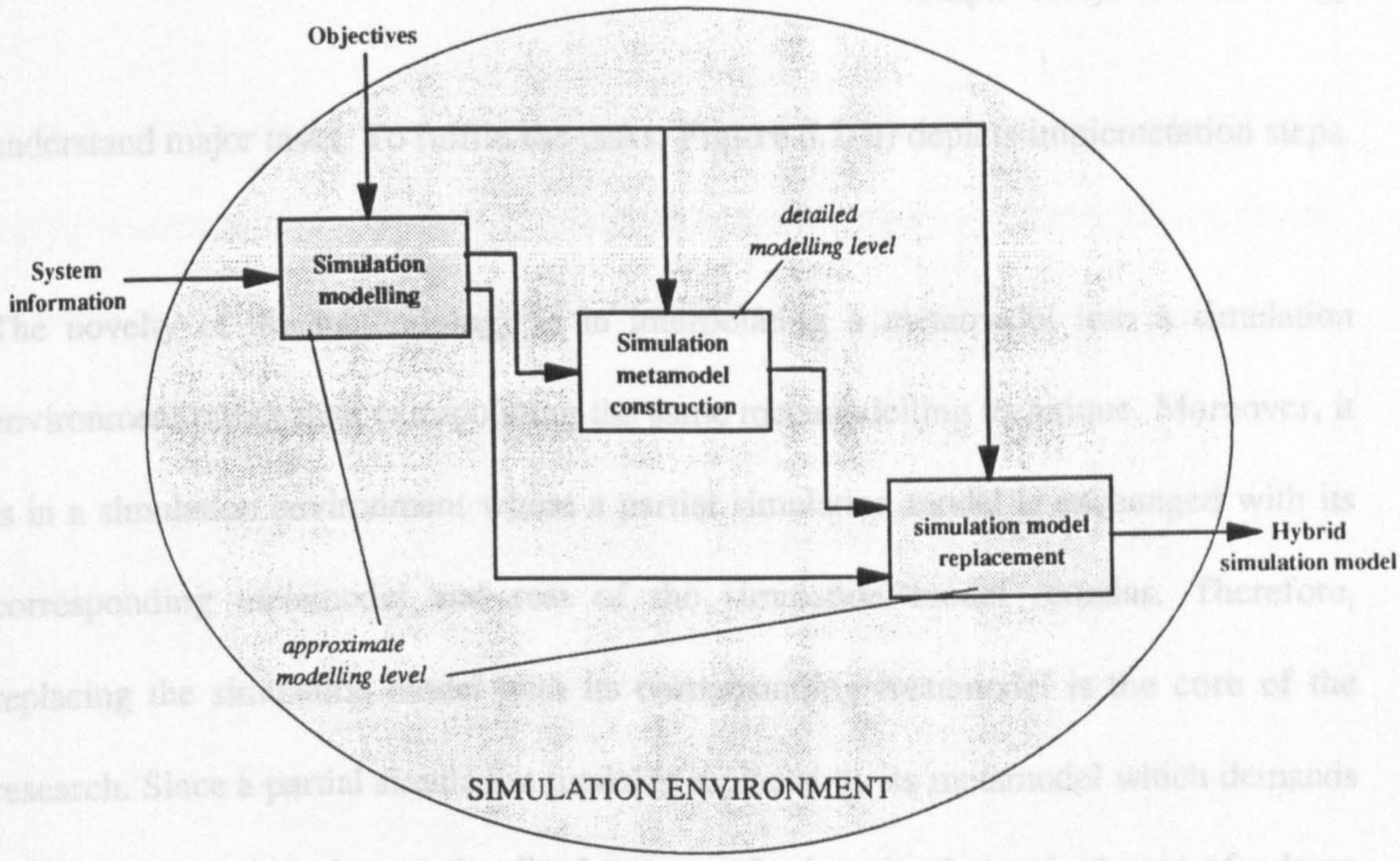
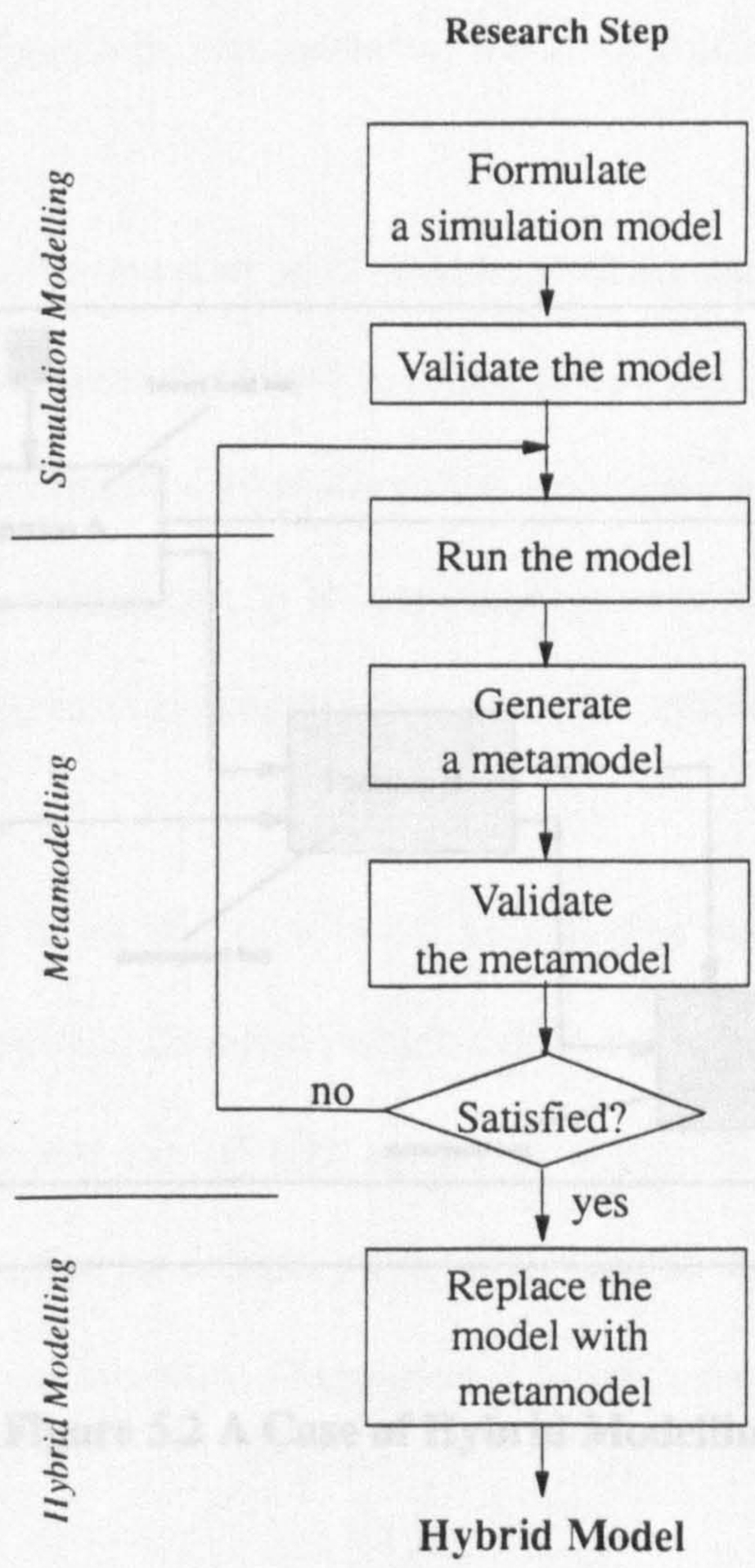



**Figure 5.1 (a) Overview of Major Research Tasks**



understand major tasks. To fulfile the tasks, **Figure 5.1(b)** depicts implemetation steps.

The novelty of the methodology is in interpolating a metamodel into a simulation environment rather than extrapolating the same metamodelling technique. Moreover, it is in a simulation environment where a partial simulation model is exchanged with its corresponding metamodel and rest of the simulation model remains. Therefore, replacing the simulation model with its corresponding metamodel is the core of the research. Since a partial simulation model is replaced by its metamodel which demands less computing time, hence, the disadvantage of using simulation in design of a large system model can be overcome. By doing so a hybrid simulation model, expressed in **Figure 5.2**, based on a simulation model built with IDEM tool shown in Figure 4.3, is produced.

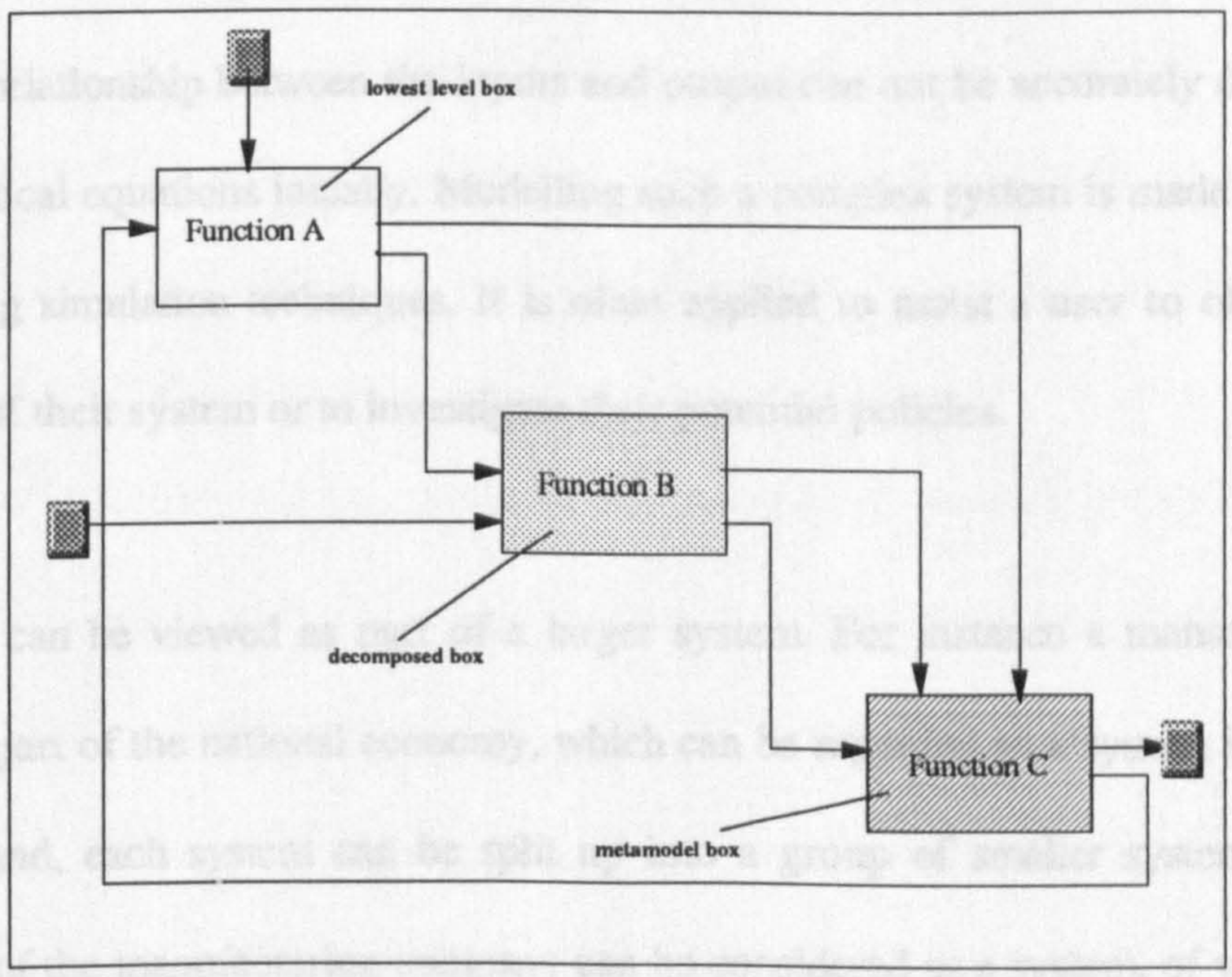



**Figure 5.2 A Case of Hybrid Modelling**

Figure 5.1 shows clearly that most of these research activities are implemented in a simulation environment. Hence, it is necessary to review the concept and philosophy of simulation first, so that the principle of embedding a metamodel in simulation can be explained explicitly.

## 5.3 Simulation Model

In terms of modelling, a model is a simplified system which is represented by elements that are of interest. It is used as a vehicle for experimentation, often in evaluating the effects of various proposals. Obviously a model should resemble the corresponding system. A system model is defined by variables and the relationship between the variables. The relationship may be represented by means of mathematical equations.

However, for most complex systems with stochastic elements, like a manufacturing system, the relationship between the inputs and output can not be accurately described by mathematical equations initially. Modelling such a complex system is made possible by employing simulation techniques. It is often applied to assist a user to obtain the knowledge of their system or to investigate their potential policies.

Any system can be viewed as part of a larger system. For instance a manufacturing company is part of the national economy, which can be regarded as a system itself. On the other hand, each system can be split up into a group of smaller systems. So a department of the manufacturing company can be considered as a system, of which the company becomes the environment. Comparing a larger system with a smaller one,

there is a commonality between them, that is transforming input into output over a time period. Hence, any system can be regarded as a simple input-output system, shown in **Figure 5.3**, regardless of their size and complexity.

**Figure 5.3 An Example of Input-Output Model**

A system consists of two types of elements, inputs and outputs. The elements of a system are represented by the so-called endogenous variables. They are characterised by their attributes. Each attribute may have a logical or a numerical value. The inputs of the system are the exogenous variables. Between these elements there are their relationships. The relationships, represented by either mathematical equations or simulation model, show how the variables react to changes in other endorgeous and exogenous variables. But relationship do not have to be limited to elements within the system, they can also refer to the environment of a larger system. Another characteristic of the system is that, over a time period, the input variables X have been transferred into the output variable Y. The principle of a transformation box is shown

in **Figure 5.4**, revealing that a transformation activity is accomplished over a time-scale and the transformation function works like a queuing system.

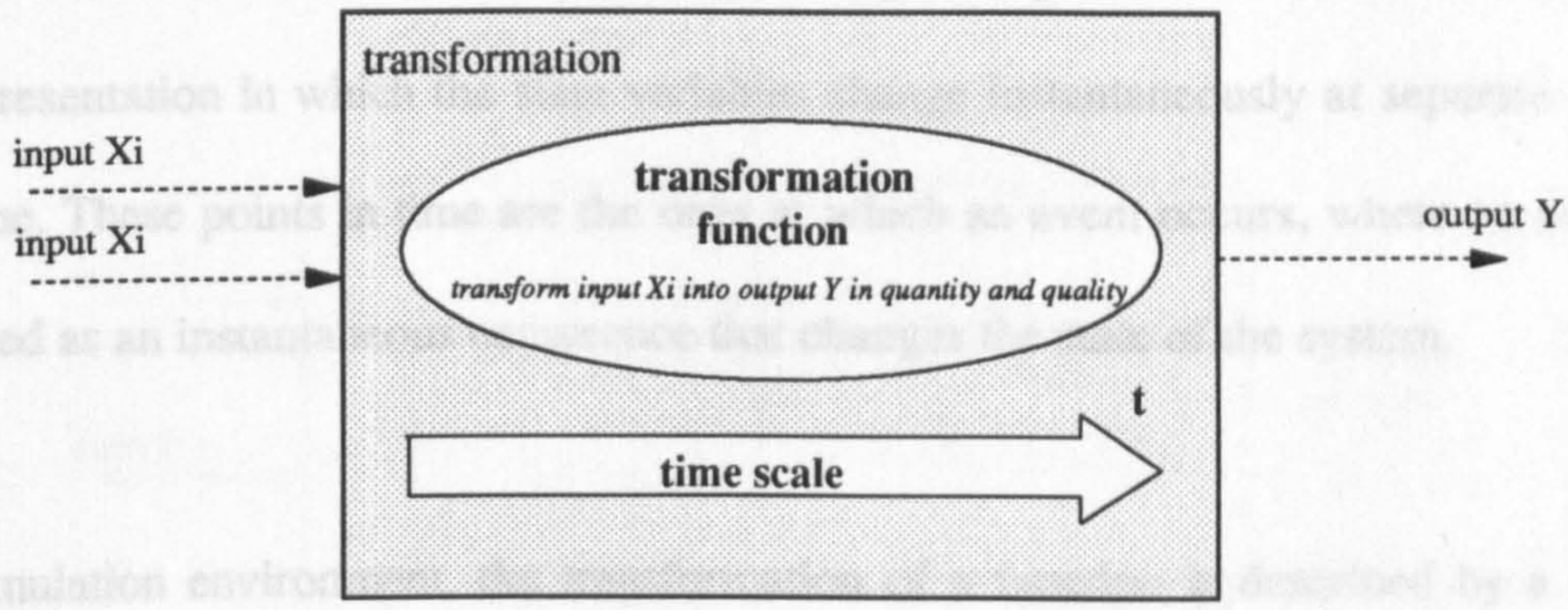



**Figure 5.4 An Illustration of Insight of Transformation**

Let's consider a service facility with a single server e.g. an information desk at a bank, the expected average delay in queue of arriving customers is estimated, where the delay in queue of a customer is the length of the time interval from the moment of his arrival at the facility to the instant he begins to be served. In order to estimate the average delay of a customer, the state variables for a discrete-event simulation model of the facility would be the status of the server, i.e., either idle or busy, the number of customers waiting in an queue to be served (if any), and the arrival time of each person waiting in queue. The status of the server is needed to determine, upon a customer's arrival, whether the customer can be served immediately or must join the end of the queue. When the server completes serving a customer, the number of customers in the queue is used to determine whether the server will become idle or begin serving the

first customer in the queue. The time of arrival of a customer is needed to calculate his delay in the queue, which is the time he begins to be served minus his time of arrival.

Discrete-event simulation concerns the modelling of a system as it evolves over time by a representation in which the state variables change instantaneously at separate points in time. These points in time are the ones at which an event occurs, where an event is defined as an instantaneous occurrence that changes the state of the system.

In simulation environment, the transformation of a function is described by a set of experimentation rather than mathematical equation(s). The simulation model is evaluated numerically, and data from the results of the simulation model are examined to produce the knowledge of the input-output system. Because the knowledge is obtained from experimentation, the technique requires a large amount of time in order to make the results reliable and accurate.

### 5.3.1 Mechanism in IDEM Simulation

After discussing simulation in general, now it is necessary to analyse the IDEM software, because the IDEM is chosen to construct a simulation model and is the environment in which a prototype tool will be developed (see chapter 4.3).

Since the IDEM offers the user a dynamic view to simulate the system (see chapter 4.2), the software provides dynamic objects, IDEM box and IDEM node[1]. Each

[1] IDEM node implies either stage-process or router-process. They are developed entirely by the author and can be applied in detailed modelling level only.

dynamic object represents an entity i.e. a system, a function or an activity in real world. Hence, every dynamic object can be regarded as a transformation box, exposed in **Figure 5.5**.

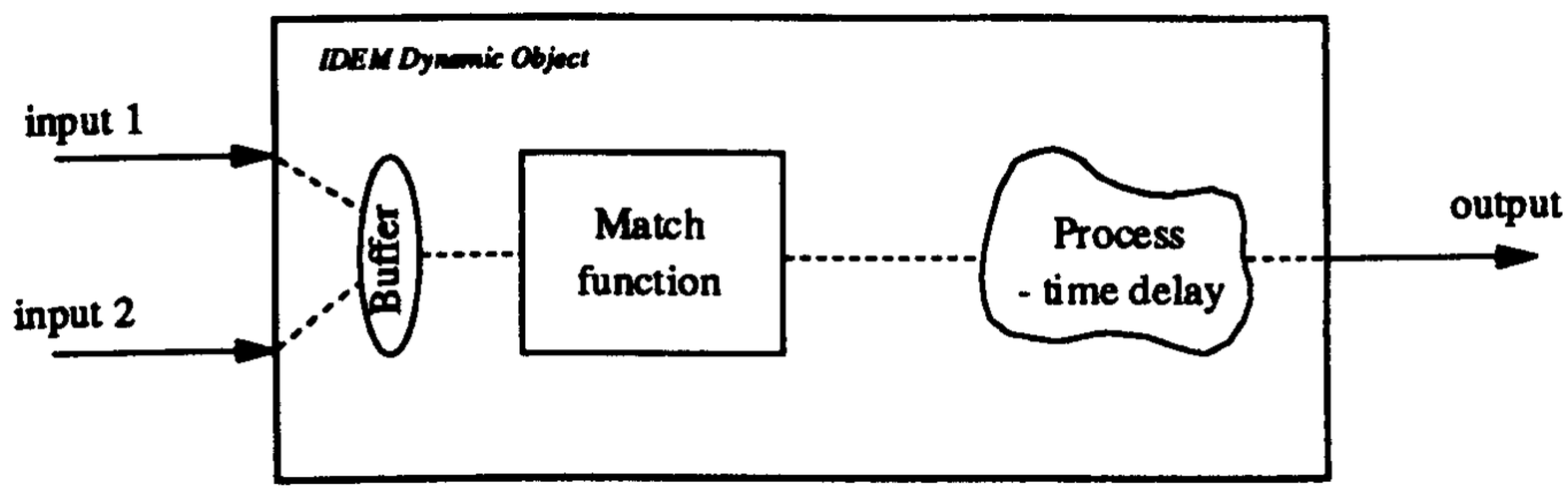



**Figure 5.5 An Illustration Of Principle of Transformation Of A IDEM Dynamic Object**

The above figure shows clearly that the transformation activity of an IDEM dynamic object comprises of match function and time elapse. The match function performs the transformation function which transforms input into output in both quantity and quality. The time delay can be seen as a process time that is required by the transformation activity. Considering an assembly job at a station, for instance, the job is assembling two parts i.e. *input1* and *input2* together. Assuming a batch of *input1* arrives at the station first, it is then put into *buffer* to wait. Once a batch of *input2* has arrived, the assembling activity, represented by *match function*, starts to transform *input1* and *input2* into product -*output* in quantity and quality. Here it is necessary to point out that time is not considered as it happens in real world because of the

philosophy of discrete event simulation. If the server[2] is idle, the job, represented by *process - time delay*, starts immediately, or say, the process time takes into account right at this point. Otherwise, the job is put into a *queue* waiting to be served[3]. As the process time is reached, the output denoted by an event in discrete event simulation is released.

A detailed IDEM model results from decomposing an IDEM box. Because the detailed model comes from the box, the matching function of the box is divided into a set of IDEM nodes which fulfil the same purpose before it was decomposed. The time is distributed among the nodes.

### 5.3.2 Function of An IDEM Box vs. Function of A Detailed Modelling Page

See **Figure 5.5** again, the dynamic node can be treated as an IDEM box.

The total transformation time can be expressed in following form:

$$t_t = t_w + t_{op} \tag{5.1}$$

$t_{op}$ denotes operation time required by a job. $t_w$ denotes waiting time. Further, $t_{op}$ is specified once a type of job is determined, and $t_w$ varies with length of queue. Thus, $t_{op}$ and $t_w$ can also be expressed as a function as below:

---

[2] The number of servers of an IDEM node is specified as 1.

[3] The queue discipline is an attribute of an IDEM dynamic object. It can be assigned the same to all dynamic objects applied in a model or assigned into the rules of each IDEM dynamic object individually.

$$t_w = f(Q_l) \tag{5.2}$$

$$t_{op} = f(type_{job}) \tag{5.3}$$

thus, $t_l$ may also be written, from (5.2) and (5.3)

$$t_l = f(Q_l) + f(type_{job}) \tag{5.4}$$

It is clear that a transformation of a IDEM box, denoting a system, is a function of the length of queue in the box, assuming the system have fixed types of job.

In contrast with the function of an IDEM box discussed above, the principle of a detailed IDEM model is exposed as follow. **Figure 5.6** is an illustration of a detailed model, which is given by breaking down an IDEM box

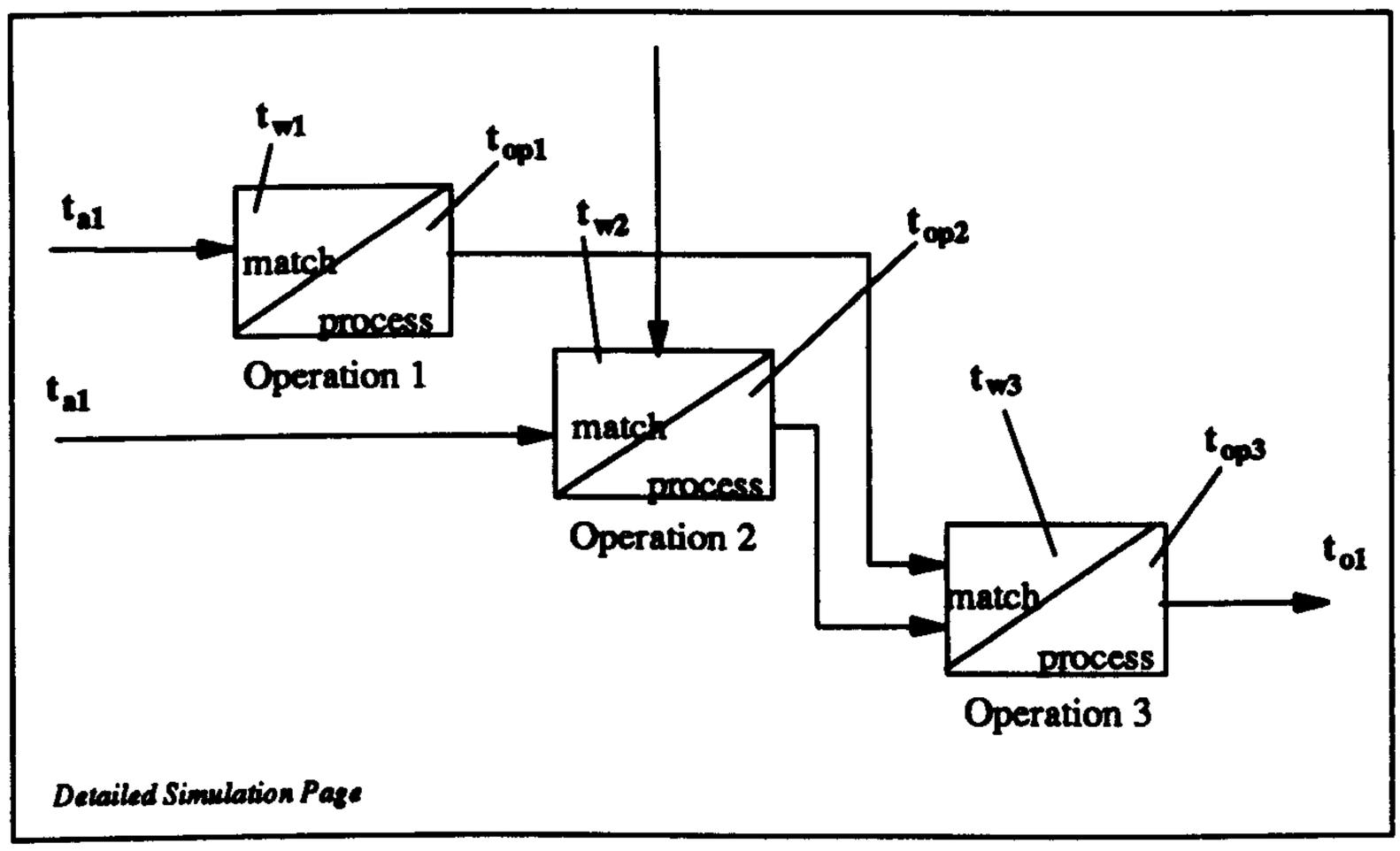



**Figure 5.6 A Illustration Of A Detailed IDEM**

In the figure 5.6, there are three objects for three different operations - operation i (i = 1, 2,3). Each object can be a machine or a shop, and it can be considered as an input-output transformation system in terms of having the functions - match and process. As a job arrives in the system, the lead time of the job is:

$$
\begin{aligned}
t_t &= t_{w1} + t_{op1} + t_{w2} + t_{op2} + t_{w3} + t_{op3} \\
&= (t_{w1} + t_{w2} + t_{w3}) + (t_{op1} + t_{op2} + t_{op3}) \qquad (5.5) \\
&= T_w + T_{op}
\end{aligned}
$$

Regarding to (5.3), $t_{opi}(i = 1,2,3)$ is a function of a job type. Thus, for a specified job, $T_{op}$ is constant. Refer to (5.2), $t_{wi}(i = 1,2,3)$ is a function of the length of a queue of the object, hence its expression is like $t_{wi} = f(Q_{il})$ ($i$=1,2,3). At a point in time, $T_w$ depends on the sum of each individual queue length. In other words, $T_w$ is a function of total number of jobs in the system. Hence, (5.5) is converted into

$$t_l = f(Q_k) + f(type_{job}) \tag{5.6}$$

If the detailed model is viewed as a whole, (5.6) is equivalent to (5.4).

In summary, a lead time varies with type of jobs and number of jobs in the queue of a system.

## 5.4 Embedding a Metamodel in Simulation

The objective of the research is to solve simulation problem by means of computing time and cost. This can only be achieved by decreasing simulation model execution time, because the simulation is a function of time and transformation (in quantity and quality). A metamodel consumes much less computing time to generate almost the same results as simulation does. Therefore, replacing a partial simulation model with metamodel is the way to minimize the identified simulation problems.

In conventional terms, a metamodel is used to interpret and analyse a simulation model. It is also often used to explain the relationship between endogenous and exogenous variables in a transformation function, ellipse part displayed in **Figure 5.4**. Here, the metamodel differs from the conventional one. It considers time only by means of the time scale shown in **Figure 5.4**. Therefore, both independent and dependent variables of a metamodel are related to time.

Before detailing the philosophy of the embedding a metamodel in the IDEM

the requirements of implanting a metamodel in a general simulation environment are discussed.

### 5.4.1 Requirements of Embedding a Metamodel in Simulation Environment

Since an embedded metamodel will replace its corresponding simulation model, it should a) perform the same function as the corresponding simulation model does, b) generate an event after a lagged interval which is the same as time spent to fulfil the transformation activity in the corresponding model. To clarify this further, an embedded metamodel should meet following requirements:

- The key requirement of embedding a metamodel in simulation is the time elapse. This requires the embedded metamodel being able to generate a reliable elapsed time.

- An embedded metamodel should possess a transformation function and a time elapse function. Because it represents an input-output system model, both functions are essential to an input-output system.

- Since a transformation function consists of transforming input to output in quantity and quality, the embedded metamodel should result in the same quantity as the simulation does.

- Since transformation function is also measured in quality, therefore, an embedded

metamodel must meet this requirement. This means that the embedded metamodel must transform inputs into correct output, in simulation terms, the output event should be produced and should carry correct information .

### 5.4.2 Embedding a Metamodel in the IDEM Simulation Environment

The above section illustrates the generic requirements of embedding a metamodel in a simulation environment. In order to embed a metamodel in the IDEM environment, it is essential that a metamodel should meet all the generic requirements. In addition, there is a need to specify other requirements which are specific only to the IDEM environment, because this research is conducted in the IDEM environment. In the IDEM environment, embedding a metamodel in simulation means replacing a decomposed page box with a metamodel box.

The generic requirements in the IDEM environment can be interpreted as:

- As described before, the matching function performs the same as the transformation function does, and the process time function performs the same as the time elapse function does in the IDEM software. Therefore, a metamodel box must have matching function and process time function.

- In the IDEM environment, a dynamic flow is the output of a box. So, a metamodel box must produce flow(s) in quantity.

- The quality of an output implies the carried concept of the output in the IDEM environment. Hence, a metamodel box must produce dynamic flow(s) on which all the necessary information are contained.

- A metamodel box must generate dynamic flow(s) after assigned elapse time .

According to the above points, the principle of a metamodel box is exhibited in **Figure 5.7.**

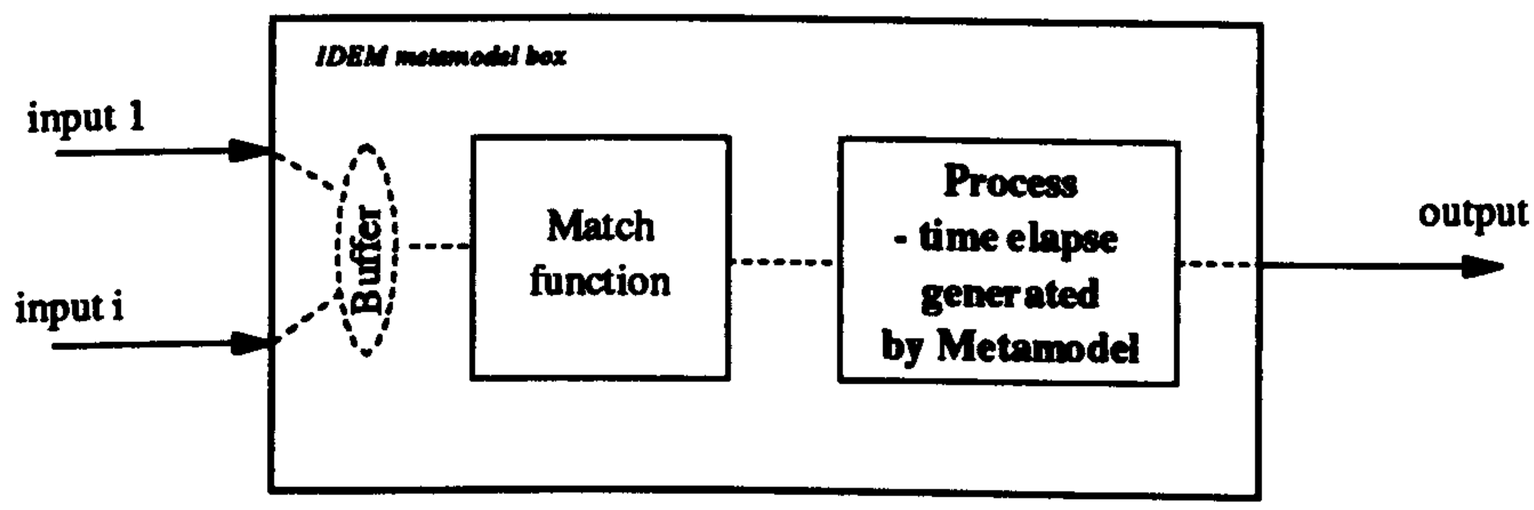



**Figure 5.7 An Illustration Of Principle of Transformation Of A IDEM Metamodel Box**

Because a metamodel takes part in the IDEM methodology, a hybrid model in which a metamodel box is used to represent a metamodel should also meet the need for multi-view. Hence, special requirements are clarified:

- To meet the need for the information view. In the IDEM environment, the

information view means a defined information structure (see chapter 4.4). So a metamodel box must have an information structure.

- To meet the need for the dynamic view. The dynamic view is described by some attributes, such as data handling function, package selection function and rules which a user applies specially. So, a metamodel must have the attributes.

## 5.5 Metamodelling

The requirements of embedding a metamodel in simulation have been discussed. The details of how to generate such a metamodel will be demonstrated below.

The research of the relevant literature (refer to chapter 3.5) reveals that many authors advocate the use of a metamodel in simulation. It has been applied for estimation of factor interactions of a system, control, system optimisation, and so on (Bitran and Chang,1987) (Li and Chiu,1993). These metamodels are often regression models (a few metamodels are piecewise linear functions and inverse polynomials).

### 5.5.1 General Regression Metamodel

The definition of the simulation metamodel is introduced as follows:

Let $X_j$ denote a factor j which influences the output of the real world system (j=1,2,3...r). A factor may be qualitative (e.g. the priority of rules chosen at given work

centres) or quantitative (e.g., the number of machines or operators available, the expected processing times on given work centres and the partial mix). Normally, the system has multiple outputs. At present, we can concentrate on single response variables. Let Y denote the system response variable of interest, which may be qualitative or quantitative (e.g., the time required to carry out a given production program). The response variable Y is a function of the factors:

$$Y = f_1(X_1, X_2, \ldots X_r) \tag{5.7}$$

This system is approximated by a simulation model. In this case, Y is a function, say $f_2$ of s factors $X_j$ (j=1,2,3,...s), plus a vector of random numbers ν, or:

$$Y = f_2(X_1, X_2, \ldots\ldots X_s) + \nu \tag{5.8}$$

where s is much smaller than the unknown s and ν which symbolises the effect of all factors X in (5.7) not explicitly represented in (5.8).The simulation model, represented by $f_2$, may be approximated in turn by a metamodel within a specific experimental area. Thus, if the e denotes the fitting error (the noise) which has expected value zero, a metamodel may be characterised as in (5.9):

$$Y = f_3(X_1, X_2, \ldots X_m) + e \tag{5.9}$$

The approach is summarised in **Figure 5.8.**

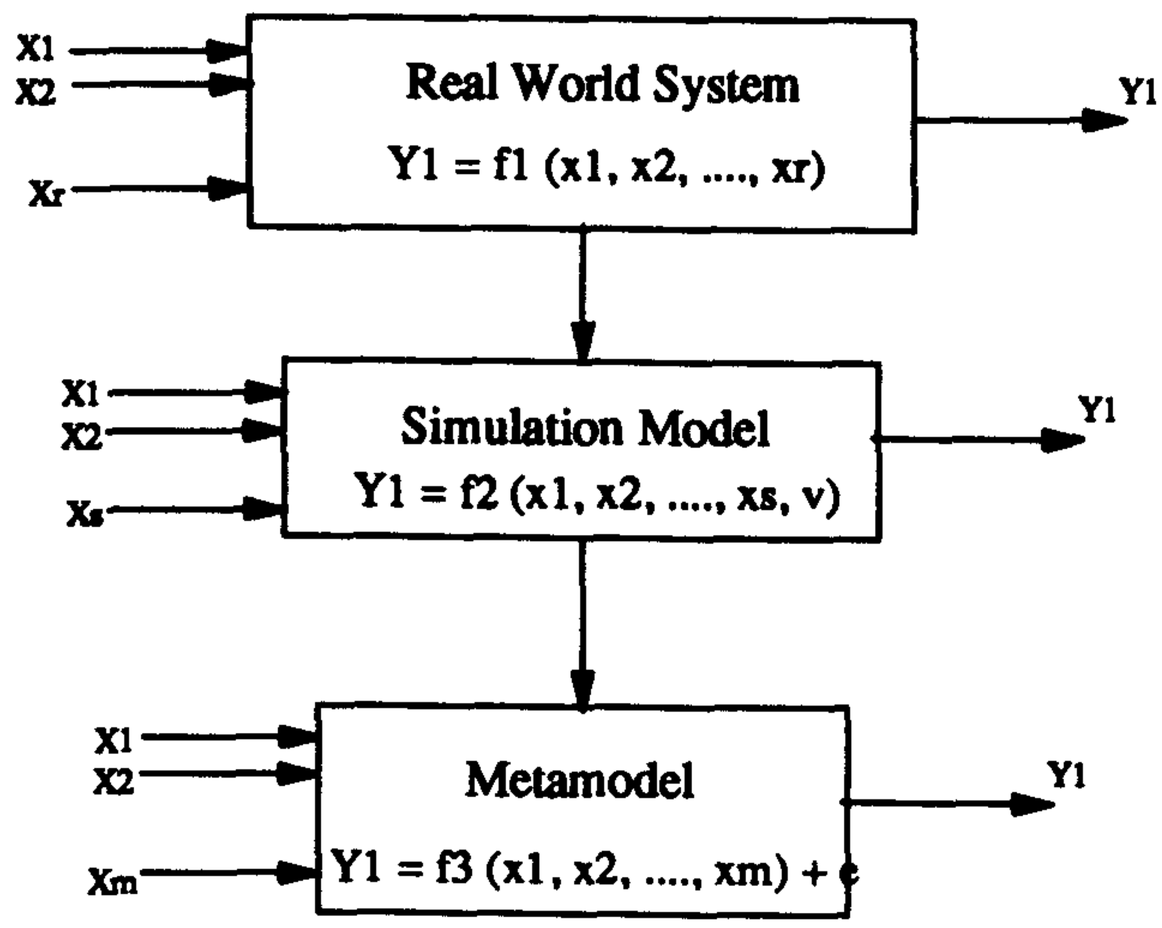



**Figure 5.8 Metamodelling**

The metamodel problem is to determine, by using the results of preliminary simulation experiments, a function $f_3$ so that for given values of the factors $X_m$, the differences between estimations of Y using the simulation model and estimation of Y using the metamodel are as small as possible.

Keijing (1979, 1987) demonstrates the application of linear regression techniques. When Y is quantitative, a first-order metamodel to express the effects of the s factors would be:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_s X_s + e \qquad (5.10)$$

Least Square is a mathematical, not a statistical problem formula. If the curve is linear

in the parameters $\beta$, we can find the least square values $\hat{\beta}$ in any textbook on regression analysis:

$$\hat{\beta} = (X'X)^{-1}X'Y \tag{5.11}$$

Under the classic assumptions, the simplest statistical model results that the errors e are normally and independently distributed with zero means and constant variances $\sigma^2$. Based on these assumptions, the least squares estimator $\beta$ is the best (minimum variance) linear unbiased estimator and $\beta$ can be tested through t and F tests.

Ideally it is preferred that an estimation of $\sigma^2$ is not dependent upon the adequacy of the fitted model. This is only possible when there are several observations on Y for at least one value of X or when prior information concerning $\sigma^2$ is available. When this approach can not be used, the estimation of $\sigma^2$ is obtained from the residual (RSS) or error sum of squares. An unbiased estimator of $\hat{\sigma}^2$ is

$$\hat{\sigma}^2 = MSE = \frac{SSE}{n-k-1} = \frac{RSS}{n-k-1} \tag{5.12}$$

in which SSE, stands for sum of square error, is calculated

$$\begin{aligned} SSE &= \sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \\ &= Y'Y - \hat{\beta}'X'Y \\ &= \sum_{i=1}^{n} Y_i^2 - \hat{\beta}'X'Y \end{aligned} \tag{5.13}$$

## 5.5.2 Metamodel Validation

Once a metamodel has been constructed, it can be used immediately. However, care must be taken to ensure that it does not generate misleading or inadequate results. To ensure the returned metamodel is trustworthy, validation is the most important step. Several methods can be found, such as split-up, double-cross and others (Montgomery, 1992) (Beck, 1984). In this research, double-cross method has been selected.

### 5.5.2.1 Metamodel Validation Method

Double-cross method splits the simulation results into two sets that are approximately of equal size. One is saved for testing the response variable, and the other is used to construct the metamodel. Two $R^2$ values (for computing the value see the following section) are then used to determine whether the metamodel is reliable, one from the set of data selected to develop the metamodel and the other from the saved set of data. If the two $R^2$ are similar, it indicates that the correlation of dependent and independent variable is denoted appropriately. Otherwise, the metamodel needs reconstruction by applying transformation function, or even by going back to re-selecting the input combinations of the simulation model.

### 5.5.2.2 Statistical Test

Within the method, statistical tests are used for determine the metamodel reliability. In

addition, $R^2$ and two other statistical tests are used, that are *F*- statistic and *t*-statistic.

- To measure the overall model utility, the multiple coefficient of determination, denoted by $R^2$, is calculated by either of the following two equations:

$$R^2 = \frac{ExplainedVariation}{TotalVariation} = \frac{SSR}{SST} \tag{5.14a}$$

or

$$R^2 = 1 - \frac{UnexplainedVariation}{TotalVariation} = 1 - \frac{SSE}{SST} \tag{5.14b}$$

Because $0 \le SSE \le SST$, it follows that $0 \le R^2 \le 1$. A value of $R^2$ that is close to 1 implies that most of the variability in Y is explained by the regression model - the metamodel. However, since adding an unimportant independent variable will increase $R^2$ to some extent, it may decrease the usefulness of the model. Therefore, it is sometimes useful to correct this by reducing $R^2$ appropriately. By doing this, a adjusted $R^2$, denoted by $R^2_{-sq(adj)}$, is defined as follows:

$$R^2_{-sq(adj)} = \left( 1 - \frac{\frac{SSE}{DF_{(ERROR)}}}{\frac{SST}{DF_{(TOTAL)}}} \right)^2 \tag{5.15}$$

Explained Variable (SSR) is computed:

$$SSR = \sum_{i=1}^{n} (y_i - \bar{y})^2 \tag{5.16}$$

and total variation (SST) is expressed as below:

$$SST = \sum_{i=1}^{n} (y_i - \bar{y})^2 = SSR + SSE \tag{5.17}$$

here SSE presents unexplained variation.

Moreover, the statistic $R^2$ itself is not enough to determine the reliability of a metamodel. The sum of squares for regression will increase and the residual sum of squares will decrease, when a variable is added to the metamodel. The addition of a regressor also increases the variance of the fitted value $\hat{y}$, so a metamodel must be constructed carefully to include only regressors that are of real value in explaining the response.

- To help in determining the value of each of the regressors in the metamodel, hypotheses are of interest. The hypotheses for testing the significance of any individual regression coefficient, such as $\beta_j$, are

$$H_0: \beta_j = 0$$
$$H_1: \beta_j \neq 0$$

If $H_0: \beta_j = 0$ is not rejected, then it indicates that the regressor $X_j$ can be deleted from the metamodel. The statistic test for this hypothesis is

$$t_0 = \frac{\hat{\beta}_j}{S_e(\hat{\beta}_j)} \tag{5.18}$$

where $S_e(\hat{\beta}_j)$ is standard error, and computed as $S_e(\hat{\beta}_j) = \sqrt{\hat{\sigma}^2 C_{jj}}$. The null hypothesis $H_0: \beta_j = 0$

### 5.5.3 Characteristics of the Metamodel

Since a metamodel developed in this research is used as part of the simulation model rather than as a post analysis tool, it is clear that the metamodel differs from conventional ones significantly. Hence, it is necessary to address the characteristics of the metamodel:

- It models time elapse. Normally, a metamodel is used to interpret simulation results in order to gain insight on the simulation. Therefore the ordinary metamodel concerns those variables that are of interest in simulation model, where time lapse is a mechanism not a research variable. However, time lapse is included in the domain of metamodelling in this research.

- It possesses not only the characteristics of modelling time lapse but also that of an ordinary metamodel. Because the metamodel is a substitute for its corresponding simulation model, it should enable reliable simulation results to be produced. In other words, it transfers inputs into outputs in quantity and quality as its corresponding simulation model does. As it has been proved above, this is the

domain of a conventional metamodel. Put it in simple terms, the metamodel can generate reliable simulation results as long as it has the characteristics of a conventional one. Since the metamodel works as part of the simulation model, it also requires to model time elapse obviously. Hence, the metamodel has a special dual characteristic.

- In a conventional way, a metamodel was used as a post analysis tool which is applied in a separate environment. Although the main aim of the research is to replace the partial simulation model with a metamodel, the benefit of the research also lies in employing metamodelling technique consistently in the same environment - the simulation environment. Once replacement takes place, the environment is shared between metamodel and simulation model.

- A hybrid is created in which the simulation model and the metamodel are executed at the same time. When regarded as a post analysis tool in the conventional way, the metamodel is applied after execution of the simulation model. In other words, the simulation model and the metamodel are executed in sequence. However, application of the metamodel in this study produces a hybrid model in which the simulation model and the metamodel work together as a single unit, therefore, the metamodel and the simulation model are executed at the same time.

- The metamodel always follows the independent variable:

  queue length

  To generate such a metamodel, other independent variables may be coupled with

the above one, that are:

type of job

batch size

The dependent variable of the elapsed time metamodel is

time lapse.

# Chapter 6

# IMPLEMENTATION

## 6.1 Introduction

The principle of the research has already been explained in chapter 5. This chapter will examine the implementation method. This chapter starts with a review of the aim of the implementation. The implementation environment, where the experimental software is developed, is than examined. Thereafter, the implementation of the hybrid methodology is described. It is followed by a discussion in which three typical applications of the simulation cases are analysed. Finally, there is a comparison between the accuracy of the research results using the software developed by the author and those using the existing software.

Section 6.4 Implementation of Hybrid Methodology comprises three major sections which are metamodelling, hybrid modelling and interface specialisation. The metamodelling section focuses on metamodel construction, metamodel validation and metamodel replacement. In the hybrid modelling section, the method of setting the metabox in the simulation environment is discussed. To aid users in applying the hybrid methodology, the interface has been customised and described at the end of this part. Through out the chapter, an example is used to illustrate the implementation steps.

## 6.2 The Aim of the Implementation Work

The implementation has the following aims:

1. To demonstrate that the metamodel can be generated with detailed simulation results and can be used to interpret simulation results in order to identify significant independent variables. The research work includes:

   a- to design detailed simulation nodes with essential features to make zoom paradigm possible.

   b- to design a data collector for collecting simulation results.

   c- to design a metamodel generator to evolve a simulation metamodel.

2. To show that a metamodel can be embedded in a simulation environment and can be used to reduce simulation time significantly. To meet this objective, the following work has been carried out :

   a- to design a metamodel box which can replace the transition box in the IDEM software.

   b- to make the metamodel box editing utility user friendly.

A range of simulation scenarios have been identified and selected for experimentation. They are described in section 6.4.

## 6.3 Implementation Environment

### 6.3.1 The Experimental Software Development Environment

The hybrid modelling methodology is implemented with the help of a prototype software tool. The software tool has been developed using LISP and LOOPS - Lisp Object Oriented Programming System. LOOPS is a product of Venue, based on Medley, and runs on a SUN 4 workstation. Since LOOPS runs in Medley's environment therefore it has all the Medley environment facilities. They include a compiler, a debugger, a list structure editor, a text-editor, file browser, class browser and some others. Because some existing statistical packages are written for MS-DOS environment, part of the research is to develop a program for implementing the hybrid modelling methodology, written specifically for LOOPS and running under LOOPS environment.

IDEM software, which is also developed using LISP and LOOPS, is another part of the environment where the experimental software works. Because the simulation model in which the metamodel is embedded is written using LISP and LOOPS, the program written for the metamodel has to take the IDEM software structure into account.

For detailed information about the software development environment of Medley-LOOPS the readers are referred to Xerox (1988), Stefik and Bobrow (1987), Venue (1992) and Goddi and Keuneke (1992). The essential concepts of object oriented

programming techniques are given in appendix II.

### 6.3.2 Metamodel Development Environment

A metamodel is derived from simulation results. The more accurate the simulation results are, the better a metamodel's reliability is. Hence, it is preferred to generate a metamodel starting from detailed simulation levels. Because the development of the IDEM tool is to meet the needs for design of manufacturing system at approximate level (refer to chapter 4), the IDEM tool did not provide those facilities to be used for constructing a simulation model at detailed level. Therefore, it is necessary to extend the IDEM tool in detailed simulation level in order to accomplish this research. **Figure 6.1** shows the metamodelling program environment and its related environments.

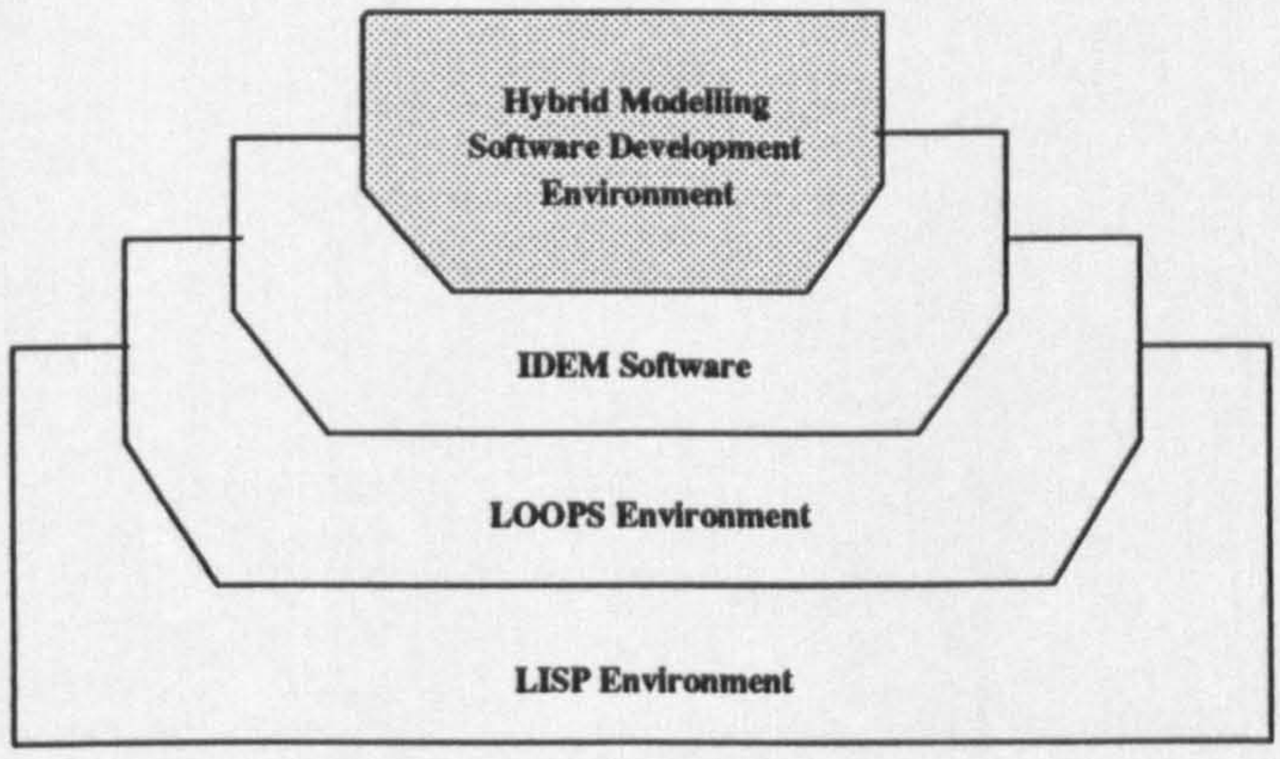



**Figure 6.1 Overall Structure of Software Developement**

### 6.3.2.1 Zoom Node Library

As illustrated in Background (see section 4.2.2), detailed simulation level is also called zoom paradigm, and the nodes used in the level are called zoom nodes. In order to construct a detailed simulation model easily, a zoom node library has been added in the IDEM tool. The concept of the library has also been identified.

Each zoom node is classified by its class structure, shown in **Figure 6.2.** The figure also reveals that a zoom node is a sub-class of the IDEM node in terms of the IDEM box. The IDEM box has a default information structure within. Therefore, the zoom node inherits the information structure from its super-class -- the IDEM box. However, each zoom node in the library has no features except its presentation image. Hence, it is necessary to specify the features of the nodes before going forward. **Figure 6.3** shows that the icon of the design-node is used in a zoom paradigm.

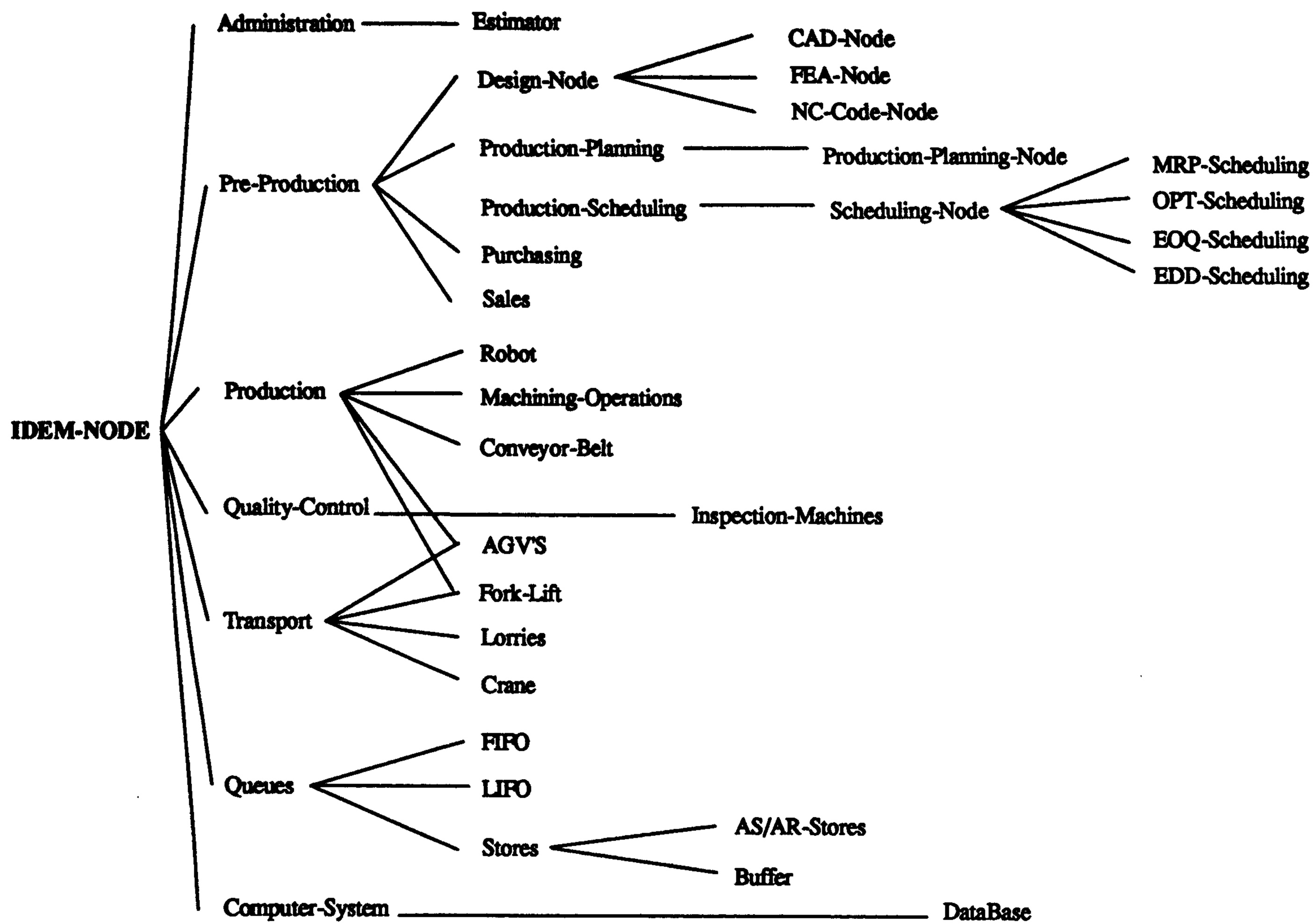



**Figure 6.2 Zoom Node Library**

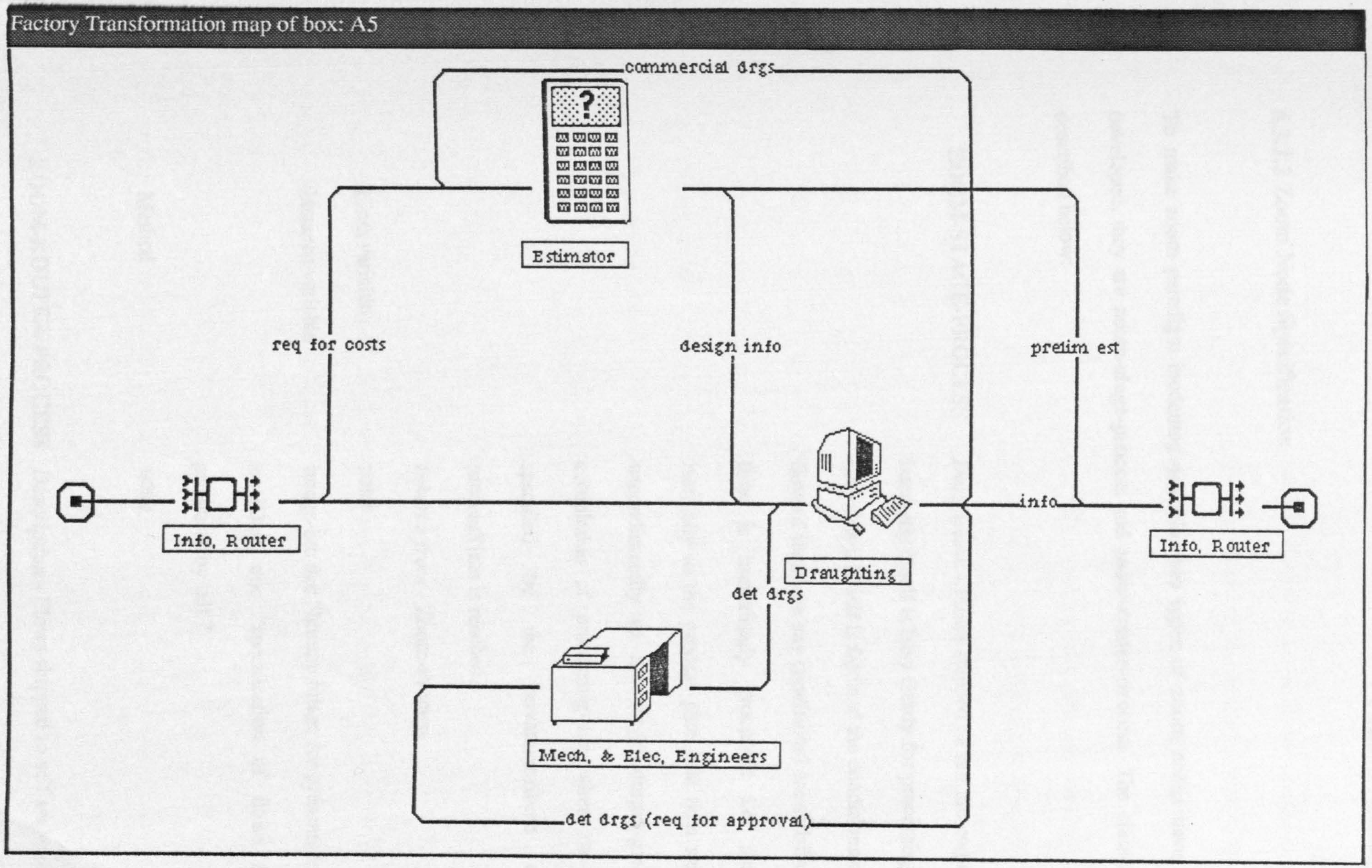



**Figure 6.3 An Example of Zoom Paradigm with Detailed Zoom Node Icons**

### 6.3.2.2 Zoom Node Specification

To make zoom paradigm modelling possible, two types of zoom nodes have been developed, they are **zoom-stage-process** and **zoom-router-process**. The classes are described below:

| | |
|---|---|
| **ZOOM-STAGE-PROCESS** | *Description* - Flows shipped to self are enqueued internally if self is busy (ready for processing at a later time). If self is free and the conditions of the flow (if there are any conditions) are satisfied the flow is immediately processed i.e. shipped internally to the process gate, and then shipped unconditionally to the single output-gate on completion of processing i.e. when the time specified by the private-attribute called processTime is reached.<br>*Inherits from* - Zoom-Process |
| Class variable | none |
| Instance variable | image-bm doc "bitmap image for presentation"<br>localData doc "specification of flows to be generated by self " |
| Method | none |
| **ZOOM-ROUTER-PROCESS** | *Description* - Flows shipped to self are enqueued |

| | |
|---|---|
| | internally if self is busy (ready for processing at a later time). If self is free and the condition of the flow is satisfied (if there are any conditions) the flow is immediately processed i.e. shipped internally to the process gate, and then shipped conditionally via a programmatically selected output gate on completion of processing i.e. when the time specified by the private-attribute called processTime is reached.<br>*Inherits from* - Zoom-Process |
| Class variable | none |
| Instance variable | image-bm doc "bitmap image for presentation"<br>localData doc "specification of flows to be generated by self " |
| Method | none |

The screen dump of their class browser is presented in **Figure 6.4**. And the icon of the two nodes is displayed in **Figure 6.5**.

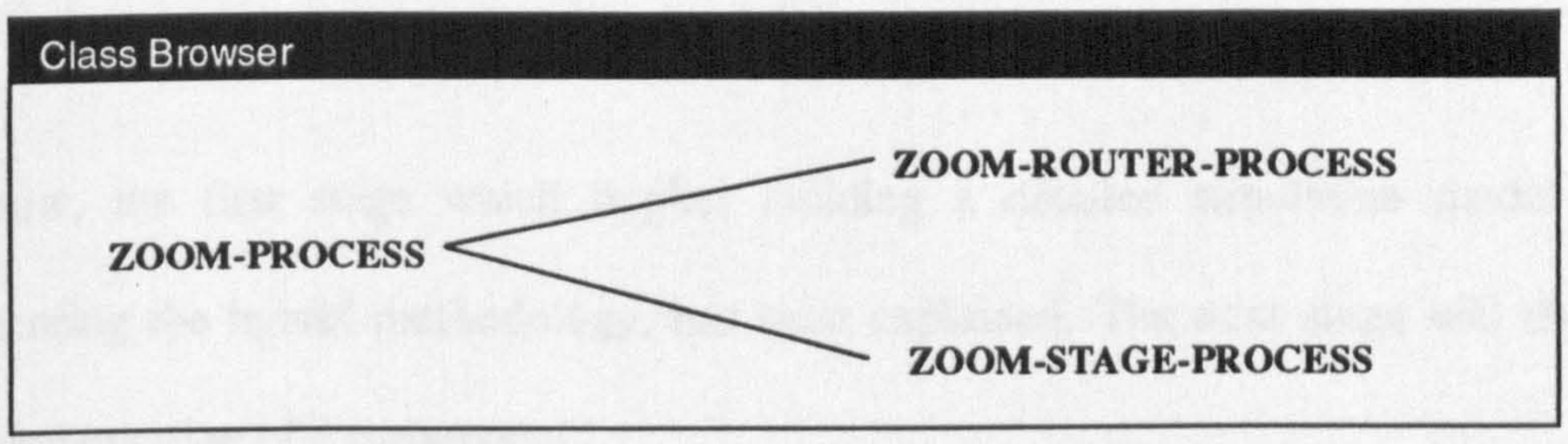



**Figure 6.4 A Screen Dump of Class Browser**

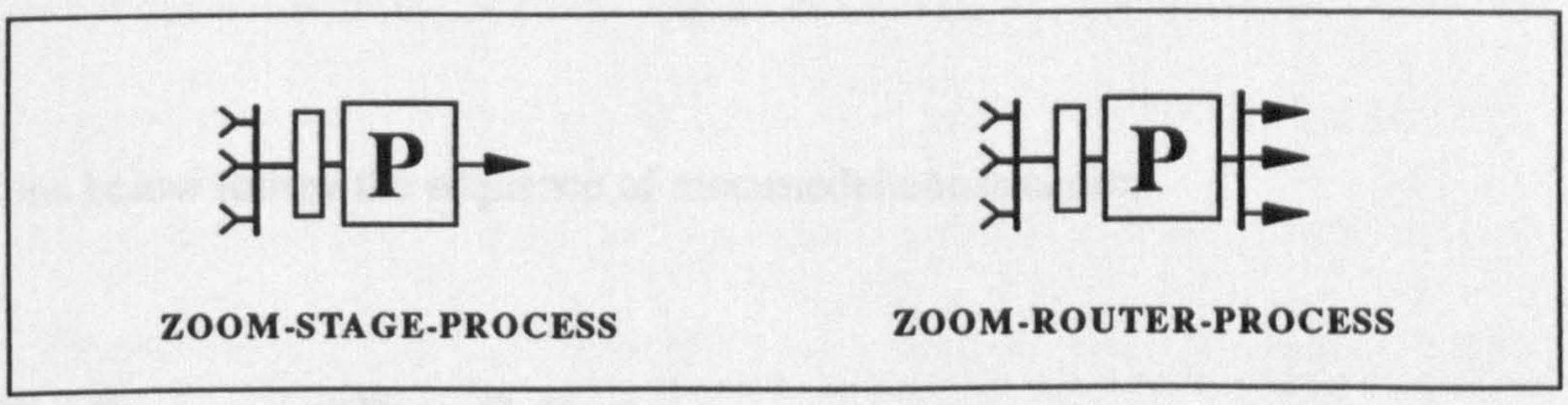



**Figure 6.5 Icon of Specialised Zoom Node**

## 6.4 The Implementation of Hybrid Methodology

### 6.4.1 The Implementation of Metamodel

Until now, the first stage which implies building a detailed simulation model of implementing the hybrid methodology, has been explained. The next stage will show the implementation of a metamodel.

The implementation is based on the concepts and principles described in chapter 5.3. Refer back to **Figure 5.8**, a metamodel is an abstract model derived from simulation results. Hence, a metamodel implementation starts from collecting simulation results. In order to implement a metamodel, a simulation data collector and a metamodel generator have already been defined and developed in the experimental software environment.

The sections below follow the sequence of metamodel construction.

#### 6.4.1.1 The Features of Data Collector

In order to gather simulation data easily and in a user friendly manner, a data collector has been designed in the experimental software. Taking the advantages of LOOPS, a data collector is defined as a sub-class of the monitor provided by the IDEM tool. Therefore, certain functions and rules, which have already been developed for the monitor, can be adopted.

The features of the data collector is classified as generic ones and specific ones. One of the generic features is to allow the user to choose a region where the user intends to apply metamodelling techniques. The region can be a whole or part of the sub-system being investigated. Then the simulation data related to independent variables and response variables selected in the region are captured.

Another generic feature is to save the data in a file if a user wishes. The file will be saved automatically in a user defined directory and the user can change their file directory whenever they like. To enable the metamodel generator to identify this type of data file, the data file name is assigned automatically with the Meta-Model-Data-File suffix. Since the date of an experiment and the independent variables selected are unique symbols of each experiment, they are saved as part of the file name.

In spite of having the above generic features, a data collector is not able to gather the right data unless it has been told what variables to monitor. Therefore, the selection mode for both dependent and independent variables is a very important feature as well. However, detailed explorations of both types of variables can vary tremendously. Hence, this feature of the data collector has been simplified to a certain degree in order to execute the embedded metamodel demonstration. The range of variables shown on the menu is also simplified to this research purpose.

The independent variable is one of these four types: they are WIP (Work In Process); WIP and Product Type; WIP and Batch Order Size; and WIP, Batch Order Size and Product Type. The dependent variables are one of the two types: they are

or finish time depending on simulation scenarios discussed in section 6.5.

#### 6.4.1.2 Data Collector Implementation

To add a data collector, the cursor is moved to the map title bar. The select button of the mouse is used to get the Title Menu, then the *Create Monitor* item of the menu is chosen to bring up the Monitor Menu and *collector* is selected from the Monitor Menu. Now a collector appears in the transition map (the detailed simulation map) and is waiting to be located in a suitable place. Before the collector is added to the map, the file directory should be set up from the LISP execution window so that the data file will be saved into the directory, otherwise, the data file will go into the current user directory.

After initialising the data collector by *Switch On*, the user is prompted to choose a region by selecting nodes and IO paths, so data from the region only will be collected. Once flows occur along those selected paths, the collector records the simulation data.

When the user initialises his simulation model, the Independent Variable Menu is displayed on the screen for the user to choose independent variable(s). But the dependent variable is determined automatically, as described before. When the user resets the simulation, the user will be asked whether to save this set of simulation results or re-select another set of objects and / or variables.

### 6.4.1.3 The Features of the Metamodel Generator

The metamodel generator have been given the following features:

a- flexible access to data files saved on hardware,

b- capability of generate a metamodel with selected data,

c- capability of validate a metamodel .

Regression analysis is used in this research to generate the metamodel. Certainly, there are many other methods which can be used such as step-wise and polynomial methods, but this is out of this research scope. At present, the metamodel generator takes first order methods to formulate a metamodel.

Another feature is also given to the generator in order to help user to identify the files easily. That is the saving of additional information associated with a data file. Because a metamodel is produced from a set of simulation results, the simulation model needs to be executed several times to create a reliable metamodel. This results in a number of data files being saved. Obviously, it is difficult to remember the conditions of each simulation experiment and to distinguish those data files. Therefore, it is necessary to add this feature.

### 6.4.1.4 Implementation of Metamodel Generator

To implement a metamodel generator, the Title Menu page (see figure 6.6) is brought

up first, then *Create Generator* is selected to create a metamodel generator instance. Then it can be added to the map where it is suitable. The system will prompt for a name to be assigned to the generator, otherwise, no name will be given if the return key is pressed at the cursor prompt. By selecting a data file, a metamodel can be generated with the set of data saved in the selected file.

To ensure the right data file is chosen, the user can check the additional information which describes the assumptions of the simulation experiment associated with the data file. This can be done by selecting *describe data file* from the Object Menu. The additional information is entered at the same time as the simulation data is saved.

Validation procedure can be invoked after the metamodel construction by choosing *Validate the Metamodel* on the Object Menu.

### 6.4.1.5 Metamodel Validation

Once the metamodel has been formulated from simulation results, the natural temptation to use it as it is can be misleading and can result in inappropriate use of the system's behaviour. Therefore, before the metamodel is used appropriately, the metamodel has to be validated.

From the many validation methods reviewed in chapter 3, the double-cross method has been chosen to validate the metamodel in this research. In this method, data is randomly split into two parts of approximately equal size. A regression model is

developed on one half, then used on the other half of the data for testing the response variable. This enables the experimenter to compute two $R^2$ values, one for the portion of observation data selected to develop the regression model and the other for the unselected hold-out portion of data. A comparison on the two $R^2$ values is then made. If these values are very different, it indicates that the metamodel is probably not valid. On the other hand, similar $R^2$ values indicate that the correlation of dependant and independent variables is denoted appropriately, thus the metamodel developed on the full set of data may be used. However, the "split-halves" should pass one more test before the full-scale observation data can be used: the regression coefficients of the two regression metamodels should be checked. If they are very different, especially with different signs, the metamodel may not be reliable.

**Table 6.1** illustrates the results of the double cross validation. The 103 observations from the same example shown in the above section were randomly split into two portions with 50 and 53 respectively. In sample I, the $R^2$ value calculated on the 50 observations which were used to generate the model was 0.978, while the results of the $R^2$ value on 53 "held-out" data was 0.965. In sample II, the respective $R^2$s were 0.97 and 0.981. **Figure 6.6** shows the transition map with a metamodel generator located. By choosing *Validation Metamodel* from the metamodel generator menu validity procedure can then be invoked. Validation is based on the metamodel which was structured by all independent variables or participated independent variables.

### 6.4.2 The Implementation of Hybrid Model

Hybrid model implementation takes place in the IDEM simulation environment. To achieve the implementation target, a meta box is created at first, and then the meta box is connected to the rest part of the model. Since the hybrid model is located at an approximate modelling level, the model can also be viewed in multiple directions.

#### 6.4.2.1 The Features of the Metamodel Box

The requirements of embedding a metamodel in the IDEM simulation environment was described in chapter 5.4.2. Considering the requirements, the author defines and develops the metamodel box class under the IDEM box. The relationship between the metamodel box and its super class -- the IDEM box is represented in **Figure 6.7.**

Since both LOOPS and LISP are objected-oriented programming environments, they support the characteristic of a sub-class inheriting the structures and methods of its super-class. Therefore, the first requirement, that is a metamodel box possessing matching function and time delay function, can be satisfied.

As explained before, time processing and matching function are separated and matching function is operated based on given inputs and the software mechanism. Hence, a metamodel box can meet the second requirement without specialization because it is a child of the IDEM box.

Same as the second requirement, the metamodel box does not require a specialization

Regression Model is :

FinishTime = 16.0063 + 0.99 ATime1 + 2.31 WIP1

| Predictor | Coefficient | Stedv | t-Ratio |
|---|---|---|---|
| Intercept | 16.0063 | 0.9497 | 16.85 |
| ATime1 | 0.99 | 0.0616 | 29.20 |
| WIP1 | 2.31 | 0.6774 | 0.0014 |

R-sq = 0.98 R-sq (adj) = 0.98

**Double - Cross Validation of Metamodel**

| Predictor | Sample I / Coef | Sample II / Coef |
|---|---|---|
| Intercept | 15.99 | 16.0082 |
| ATime1 | 0.98 | 0.989 |
| WIP1 | 2.227 | 2.30 |
| | | |
| R-sq | | |
| Select | 0.978 | 0.97 |
| Un-select | 0.965 | 0.981 |

**Table 6.1 A Example of Metamodel Validation Results**

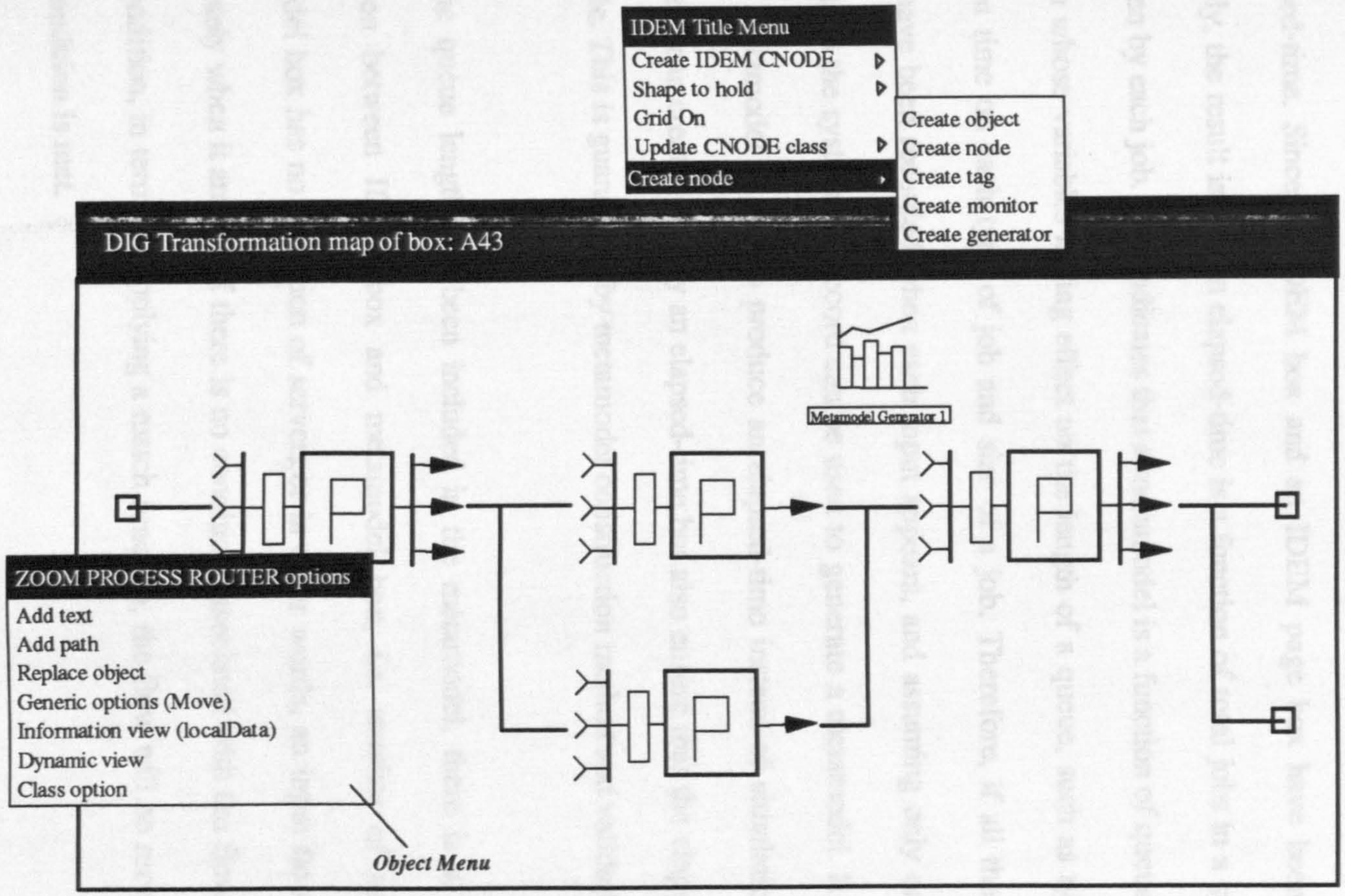



**Figure 6.6 An Example of A Detailed Model with A Metamodel Generator**

to meet the third requirement -- transforming input into output in quality. The reason has already been explained as above because it is part of the duty in matching function.

The last but not least important requirement is that a metamodel box should produce an elapsed-time. Since an IDEM box and an IDEM page box have been analysed previously, the result is that an elapsed-time is a function of total jobs in a system and time taken by each job. This indicates that a metamodel is a function of queue length of a system whose variables having effect on the length of a queue, such as type of job, operation time on each type of job and size of a job. Therefore, if all the jobs in a system have been recorded when each input appears, and assuming only one type of job exists in the system, the record can be used to generate a metamodel. It is evident that the metamodel is able to produce an elapsed-time instead of simulation. Such a metamodel can create not only an elapsed-time but also ensure that the elapsed-time is credible. This is guaranteed by metamodel construction method and validation.

Since the queue length has been included in the metamodel, there is a significant distinction between IDEM box and metamodel box, i.e. number of servers. The metamodel box has no limitation of servers,or in other words, an input flow is served immediately when it arrives, if there is no condition associated with the flow. If a flow has a condition, in terms of applying a match function, the flow will be served as long as the condition is met.

It is clear that the only concern in the metamodel box is the process time irrespective of whether there is a flow condition or not. The process time is determined by the metamodel.

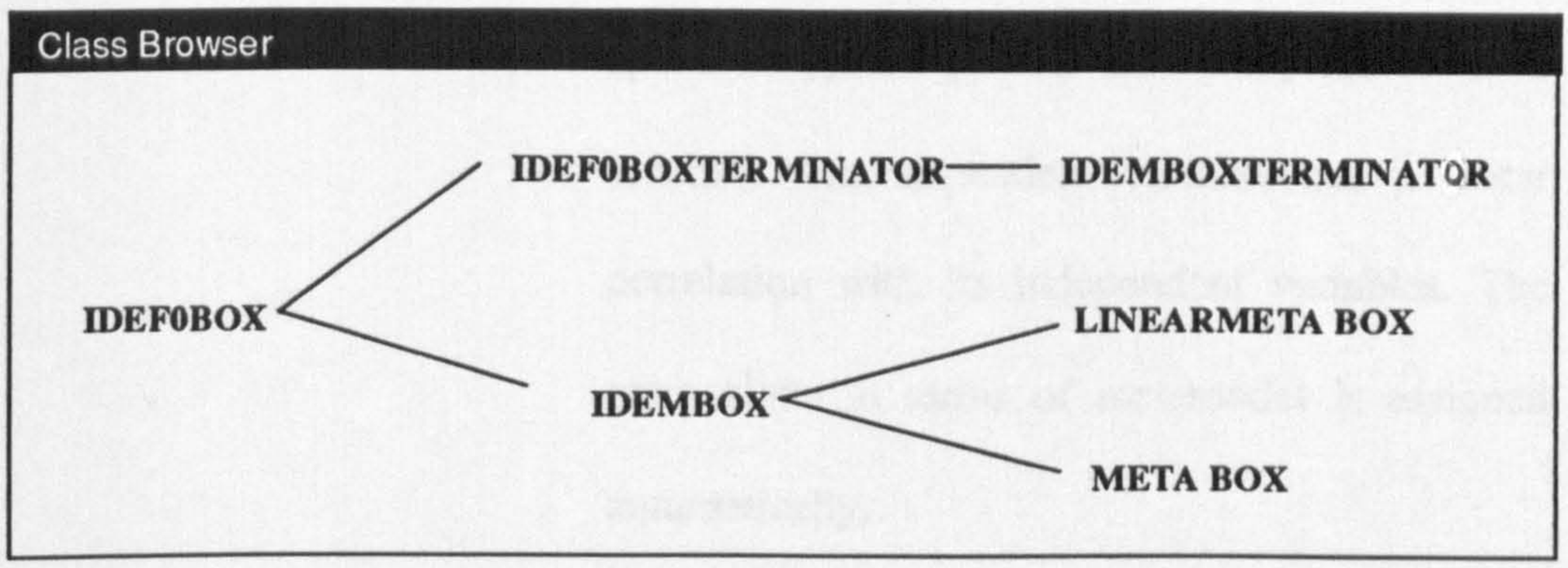



**Figure 6.7 A Screen Dump of Meta Box Class Browser**

A metamodel can vary depending on the dynamic characteristics of a simulation model and the number of independent variables identified. It is too complex to make an intelligent system which can transfer all features of a metamodel into a metamodel box automatically. Hence, only two types of metamodel box, the Linear Meta Box and the Meta Box, have been developed in the experimental software. The details of the two boxes are below:

**LINEARMETA BOX** *Description* - Flows shipped to self are immediately processed (shipped internally to the process gate) if there is no Match function, otherwise, flows shipped to self are queued to match first, and then are processed. Consequently, the flows are shipped

| | |
|---|---|
| | unconditionally to the single output-gate on completion of processing i.e. when the time specified by the private-attribute processTime is reached. The dependent variable has a linear correlation with its independent variables. The processTime in terms of metamodel is assigned automatically.<br>*Inherits from* - IDEM-BOX |
| Class variable | none |
| Instance variable | image-bm doc "bitmap image for presentation"<br>localData doc "specification of flows to be generated by self " |
| Method | none |
| **META BOX** | *Description* - Flows shipped to self are immediately processed (shipped internally to the process gate) if there is no Match function, otherwise, flows shipped to self are queued to match first, and then are processed. Consequently, the flows are shipped unconditionally to the single output-gate on completion of processing i.e. when the time specified by the private-attribute processTime is |

| | |
|---|---|
| | reached. The dependent variable does not have linear correlation with its independent variables. ProcessTime/finishTime should be edited by the user. |
| | *Inherits from* - IDEM-BOX |
| Class variable | none |
| Instance variable | image-bm doc "bitmap image for presentation" |
| | localData doc "specification of flows to be generated by self " |
| Method | none |

### 6.4.2.2 Implementation of The Metamodel Box

Once a type of metamodel box has been chosen to replace the detailed sub-system simulation, the next stage is the running of the hybrid model. The metamodel box can be linked with other parts of the simulation model before this can happen. It is necessary to point out that connection procedures fall in with IDEM simulation development steps in terms of the information and the dynamic view respectively.

#### Information View

In hybrid modelling, the information view means setting information into the metamodel box regardless the type of metamodel box. To cope with the IDEM software, the metamodel box has to use a certain structure of IDEM box, one of these

software, the metamodel box has to use a certain structure of IDEM box, one of these is localData. **Figure 6.8** is the localData of a Linear Metabox. The information structure of the box has been kept the same as that of the IDEM Box in the simulation except for the process time which is assigned by the metamodel.

SEdit Package: INTERLISP

```
((input inputQueue
        (control controlQueue)
        (mechanism1 mechanismQueue)
        NIL LiMetamodelProcessTime
        (FLOW output
                ("Finished")
                (modelA 50 II)
                NIL destination1))))
```

**Figure 6.8 Example of localData of A Linear Metabox**

To input the information into the IDEM Box, the user must name all input flows, name the gate where a corresponding flow enters, a set of conditions if there are any, as well as the output flows. Data handling function can also be assigned so that the data is able to be processed and returned as the user desires.

The information structure of a Meta Box (non-linear) is laid out in **Figure 6.9**. It is similar to the Linear Meta Box except for the process time. To set the information in it, procedures can be taken the same as those in the linear metamodel box.

```
SEdit Package: INTERLISP
((input1 inputQueue
        (control controlQueue)
        (mechanism1 mechanismQueue)
        NIL MetamodelProcessTime1
        (FLOW output
                ("Finished")
                (modelA 50 II)
                NIL destination1)))
((input2 inputQueue
        (control2 controlQueue)
        (mechanism2 mechanismQueue)
        NIL MetamodelProcessTime2
        (FLOW drawing
                ("Drawing Set")
                ((DN301 1) (DN304 1) (DN331 2))
                NIL destination2))))
```

**Figure 6.9 Example of localData of A Metamodel Box**

## Dynamic View

As soon as the information of a metamodel box has been set, the last important step is to edit the dynamic characteristics of the metamodel box. Those dynamic features include process time, flow image and flow path name. The process time is central among those features in the metamodel replacement.

Since limited independent variables have been considered in the experimental software development, the linear metamodel is very simple. Therefore, in these circumstances, all the coefficient data of the metamodel has been passed from the metamodel

generator into the metamodel box. Calculation of the process time is therefore straight forward. So the process time in the linear metamodel box is assigned automatically without editing. However, most simulation situations can not be expressed by such a simple mathematical model and that the process time can not be generated easily. The user should edit the process time by selecting the *Editing Dynamic Item* from the metamodel box Object Menu. To do this effectively and efficiently, the user is required to have some knowledge of LISP and LOOPS. **Figure 6.10** is an example of the process time editing window.

Other auxiliary editing is then done before running the hybrid simulation model. Because the metamodel box is inherited from the IDEM box, flow image a string type of image - can be edited and changed easily and conveniently as in the IDEM box. The principle of the metamodel has been described in chapter 5, interaction between the metamodel box and other objects (IDEM or IDEF0PAGE box) flow paths are illustrated by an arrow in diagram. Since the simulation controller needs to identify the correct name of a path to run the simulation model, the name of a flow path in the metamodel should be kept the same as that in the simulation model before its replacement. In other words, the path names in diagram. Since the simulation controller needs to identify the correct name of a path to run the simulation model, the name of a flow path in the metamodel should be kept the same as that in the simulation model before its replacement. In other words, the path names in the metamodel information view should be the same as those in the transition node replaced by.

```
* SEdit MetamodelProcessTime Package: INTERLISP

(LAMBAD (self localData flow)                          ; Edited 26-Oct-94 16:30
                                                       ; by bing
  "To get process time of a meta box."

  (LET ((intercept 20.8)
        (PLst (@ self parameter))
        (WIP (@ self localQLength))
        (type (CAR (@ (@ (@ flow package) datum) data)))
        (typeNumberLst (@ self typeList))
        (processTime 0))
    (SETQ processTime
          (PLUS intercept
                (PLUS (TIMES type 1.38) (TIMES WIP 2.6))))))
```

**Figure 6.10 An Example of Process Time Editing Window**

### 6.4.3 Interface Customization

Because the experimental software is developed in LOOPS, all facilities offered by LOOPS can be used for any particular application, such as the window facilities. An interface has been developed using these facilities to make implementation of embedding a metamodel easier. The interface is shown in **Figure 6.11**. The user interface is a menu driven application.

The transition map Title Menu brings up the Monitor Menu to add a particular data collector. The Object Menu of the data collector provides features for the implemented

data collector. They are used to define a metamodel. This menu also indicates where the data file has been saved. The Object Menu of the metamodel generator provides both the implemented metamodel generating feature and the metamodel validation feature. The functionality of each Icon Menu can be found in a message that appears in the black window called the "Prompt Window" whenever the icon is selected and held with the arrow cursor.

By selecting any icon in any menu, an interactive session with the user will start which enables the user to query about specific information. A prompt window of the transition map appears with a question prompt. A "Metamodel Execution Window" is also automatically opened with the output of the experimental metamodelling results. The results of metamodel validation are shown in the same window. The window can be manipulated by an additional menu attached at the top of the window. This additional menu provides facilities like to clear, close, shrink, move and scroll the windows.

In the implementation of embedding metamodel, the user identifies the independent variables in the detailed simulation model, while the data collector asks for the nodes' information to be attached and the data to be collected. Therefore the nodes and independent variables must be entered first, and then other functions of the data collector can be performed.

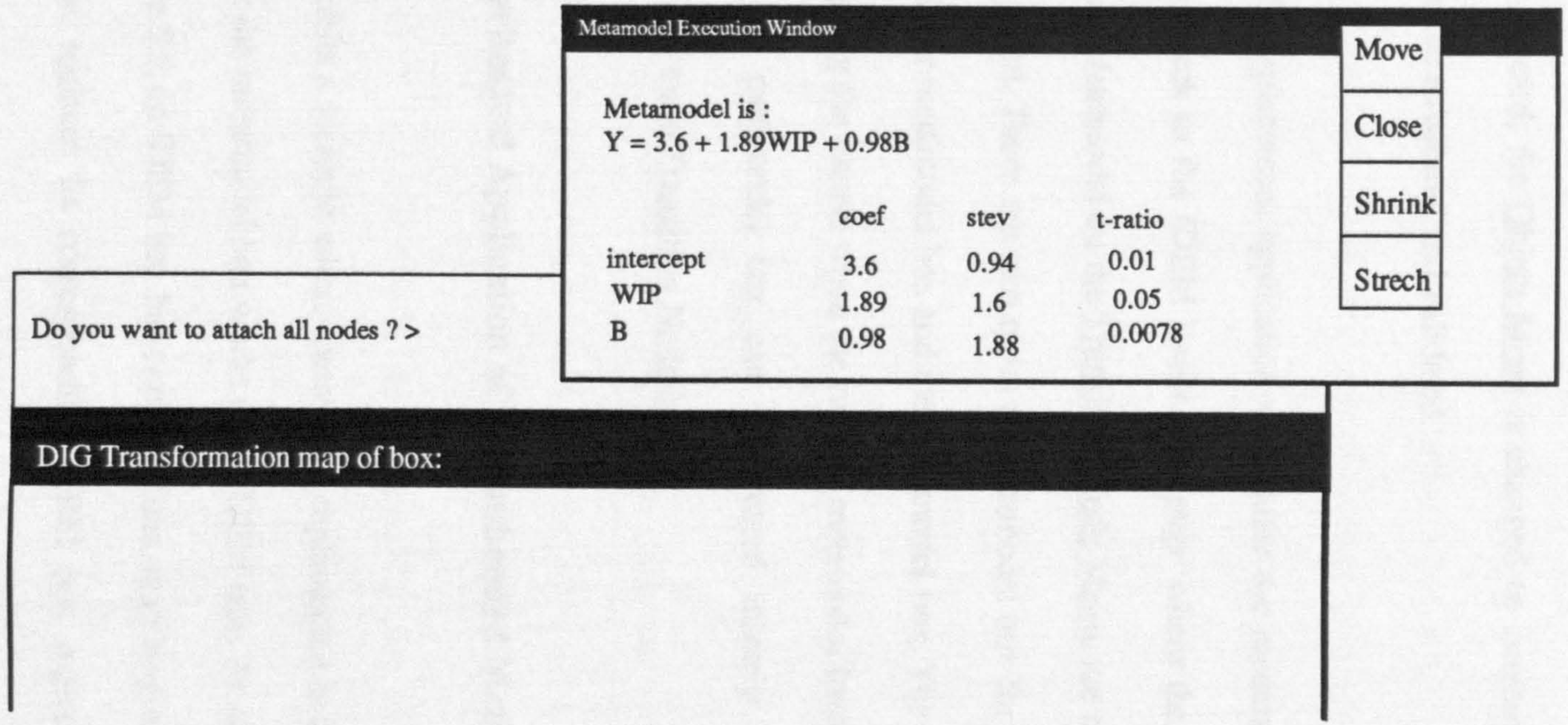



**Figure 6.11 The User Interface**

The Metamodel Generator queries the user about the simulation data saved and then it generates the corresponding metamodel by the selected simulation data. The generator lists all data files with defined file name suffix - IDEM-META-DATA. Once the data file has been selected, the Object Menu is changed to another view from where the metamodel can be structured and validated.

The metamodel replacement application starts after the metamodel validity. Now the user should go back to the IDEM lower level page where the transition node is. By picking *Replace Metamodel* on the Transition Node Menu, the metamodel replacement process is triggered. There are two types of metamodel box for replacement purposes. They are the linear metamodel box and the metamodel box. The metamodel box can be gained by dragging the mouse when the replace metamodel item has been selected and held. The linear metamodel box can be created directly by selecting *Replace Metamodel* on the main Transition Node Menu.

## 6.5 The Experimental Application of the Embedded Metamodel

Figure 5.2 exhibits a scenario where metamodel replacement is carried out. The figure has shown that the metamodel box works as an IDEM box. As mentioned in chapter 5, shown in figure 5.5, an IDEM box has two functions, matching and processing. Since a metamodel box replaces its corresponding IDEM box regardless of the way the information is arranged around the IDEM box, the replacement does not affect on matching part of the box activity. So it is only the time attributes which act as response variable should be considered.

On the other hand, an IDEM box is decomposed down to formulate a detailed simulation model where time attributes are divided into small units and distributed through the detailed model. Because the nature of input and output information for the simulation can vary extensively, the formulation of a metamodel becomes extremely complicated. During the experiments, several simulation scenarios were identified by the author. Each of them has been analysed individually in the following sections.

### 6.5.1 Design for One Input and One Output Simulation Scenario in Zoom Paradigm

The simulation scenario described here is illustrated in **Figure 6.12**. This case is the simplest situation from which a metamodel can be generated. There is one input to and one output from the sub-system, no matter how many nodes are involved and what kind of routing is related to the nodes. A node can represent a process, a machine, a cell level, or a worker in an office or in a factory. This research is only interested in correlation between input and output. Because there is only one exit from in the sub-system, all the nodes in the sub-system are involved with the process to produce the output / response. Therefore, all nodes' performance is of interest and monitored.

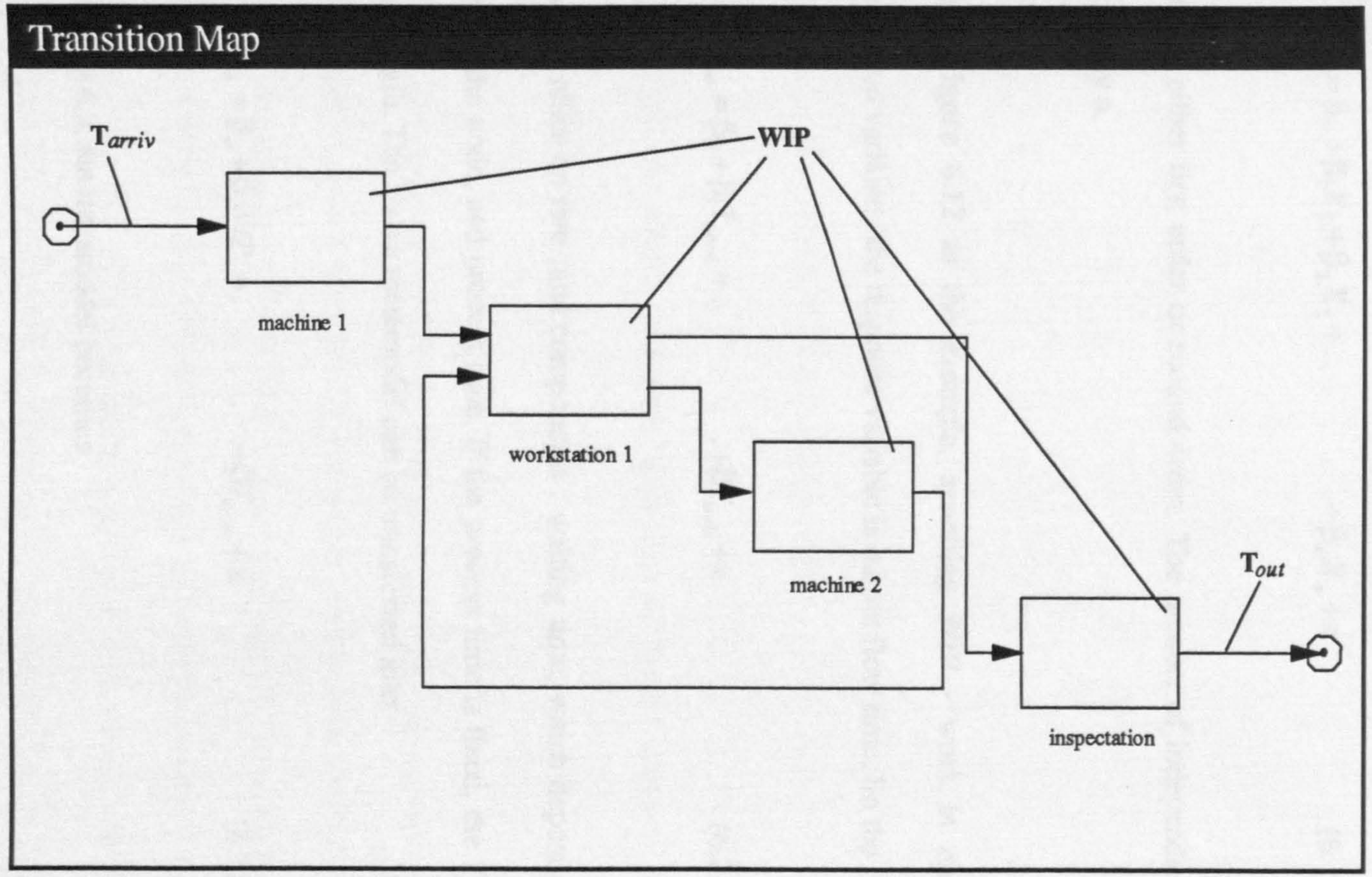



**Figure 6.12 One Input and One Output Simulation Scenario**

A metamodel is a simplest equation in terms of only nodes' performance and input flow time related to the response. The form of the metamodel may be

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + . \quad . \quad . \quad . \quad . + \beta_n X_n + \varepsilon \qquad (6.1)$$

X can be either first order or second order. The number of independent variables are denoted by n.

Taking **Figure 6.12** as the example, assuming WIP - work in process - is the independent variable, the response variable is output flow time. So the metamodel is:

$$T_{out} = \beta_0 + \beta_1 T_{procl} + . \quad . \quad . \quad . \quad . + \beta T_{arriv} + \varepsilon \qquad (6.2)$$

each $T_{proc}$ relies on two time components : waiting time, which depends on the queue length in the node, and process time. If the process time is fixed, the $T_{proc}$ varies with queue length. The above metamodel can be transferred into:

$$T_{out} = \beta_0 + \beta_1 WIP_1 + . \quad . \quad . \quad . \quad . + \beta T_{arriv} + \varepsilon \qquad (6.3)$$

to simplify 6.3, the metamodel becomes

$$T_{out} = \beta_0 + \beta_1 WIP + \beta_2 T_{arriv} + \varepsilon \qquad (6.4)$$

WIP denotes all WIP of the sub-system.

Since the regression analysis is applied only where simulation is in steady state, it is obvious that $\beta_2$ should be 1 or very close to 1. So the above equation is turned into

$$T_{out} = \beta_0 + \beta_1 WIP + T_{arriv} + \varepsilon \tag{6.5}$$

and process time in a box is calculated by

$$\begin{aligned} T_{proc} &= T_{out} - T_{arriv} \\ &= \beta_0 + \beta_1 WIP + \varepsilon \end{aligned} \tag{6.6}$$

it is clear that process time varies with WIP only, or to say, it varies with those independent variables that have an impact on WIP.

### 6.5.2 Design for Multiple Inputs and One Output Simulation Scenario in Zoom-In Paradigm

This simulation scenario is exhibited in **Figure 6.13**. The case can be analysed with the same method as above, although the metamodel seems to be more complex than the last case. For example, flow A arrives later than flow B, after each flow has been processed, eventually they will meet in node S to merge for assembly. If the flow A is much larger than flow B, it is apparent that output time varies with flow A's arrival time and operation time at node S. Therefore, the metamodel can be determined based

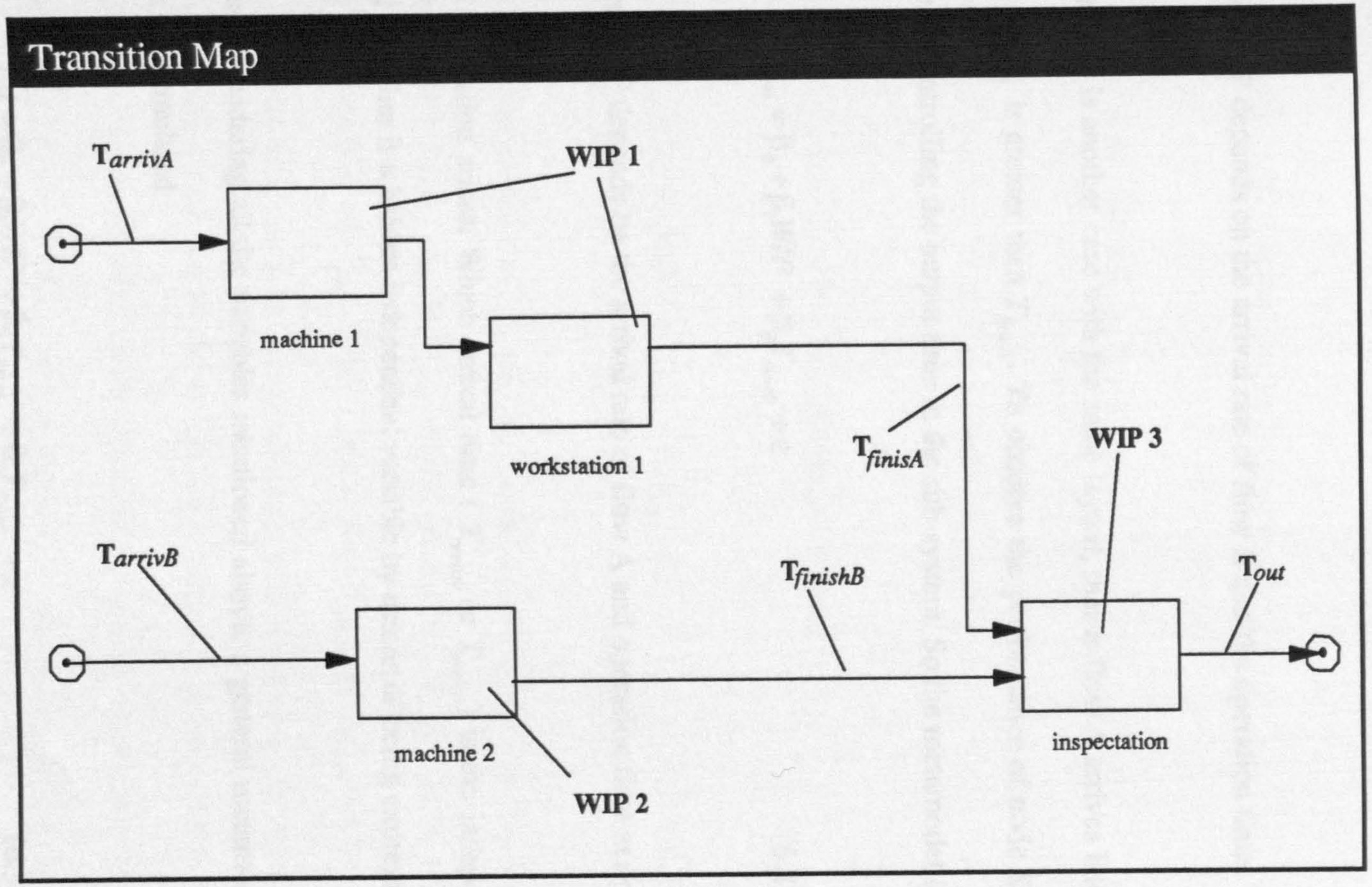



**Figure 6.13 Multiple Inputs and One Output Simulation Scenario**

on the above exploration as:

$$T_{out} = \beta_0 + \beta_1 WIP + \beta_2 T_{finisA} + \varepsilon \tag{6.7}$$

here, WIP depends on the arrival rate of flow B and the operation time.

But there is another case with the same layout, that is flow A arrives later than flow B, but $T_{finishB}$ is greater than $T_{finisA}$. To observe the performance of node S, flow B is the key to controlling the output time in the sub-system. So the metamodel looks like:

$$T_{out} = \beta_0 + \beta_1 WIP + \beta_2 T_{finisB} + \varepsilon \tag{6.8}$$

here, WIP depends on the arrival rate of flow A and operation time at node S.

So a question arises. Which arrival time ( $T_{arrivA}$ or $T_{arrivB}$) is the independent variable and the other is a hidden independent variable by means of being expressed by WIP?

After considering all the variables mentioned above, a general metamodel in this case can be formulated

$$T_{out} = \beta_0 + \beta_1 WIP + \beta_2 T_{finisA} + \beta_3 T_{finisB} + \varepsilon \tag{6.9}$$

Now, applying equation 6.6 to the above equation, the equation 6.7 is

$$
\begin{aligned}
T_{out} = & \beta_0 + \beta_1 WIP \\
& + \beta_2 (\beta_{0A} + \beta_{1A} WIP_A + \beta_{2A} T_{arrivA} + \varepsilon_A) \\
& + \beta_3 (\beta_{0B} + \beta_{1B} WIP_B + \beta_{2B} T_{arrivB} + \varepsilon_B) \\
& + \varepsilon
\end{aligned}
\tag{6.10}
$$

to re-organise 6.10, it turns into the following form:

$$
\begin{aligned}
T_{out} = & (\beta_0 + \beta_2 \times \beta_{0A} + \beta_3 \times \beta_{0B}) \\
& + \beta_1 WIP + \beta_2 \times \beta_{1A} WIP_A + \beta_3 \times \beta_{1B} WIP_B \\
& + \beta_2 \times \beta_{2A} T_{arrivA} + \beta_3 \times \beta_{2B} T_{arrivB} \\
& + (\varepsilon + \varepsilon_A + \varepsilon_B)
\end{aligned}
\tag{6.11}
$$

hence, the equation 6.11 can be simplified to become

$$
T_{out} = \alpha_0 + \alpha_1 WIP + \alpha_2 T_{arrivA} + \alpha_3 T_{arrivB} + \delta \tag{6.12}
$$

as mentioned above, both $\alpha_2$ and $\alpha_3$ should equal 1 or be very close to 1. If $T_{arrivA}$ is much greater than $T_{arrivB}$, $\alpha_3 T_{arrivB}$ can be omitted and $T_{arrivB}$ can be considered that it has effect on WIP. If $T_{arrivB}$ is much greater than $T_{arrivA}$, $\alpha_2 T_{arrivA}$ can not be omitted and $T_{arrivA}$ should be considered having effect on WIP. If flow A and flow B both appear almost at same time at node S, though the matching function of a box still depends on the latest flow arrival time, either of them can be used to generate metamodel within tolerance. In this case WIP varies almost with operation time in the node. It is obvious that this equation has the same elements as simplest case described in the last scenario. The operation time in this scenario is:

$$T_{proc} = T_{out} - T_{arrivA} = \alpha_0 + \alpha_1 WIP + \varepsilon \qquad (6.13\ a)$$

or

$$T_{proc} = T_{out} - T_{arrivB} = \phi_0 + \phi_1 WIP + \tau \qquad (6.13\ b)$$

### 6.5.3 Design for Multiple Inputs and Multiple Outputs Simulation Scenario in Zoom-In Paradigm

There is another case which often happens in the IDEM simulation model. That is a multiple inputs and multiple outputs sub-system, shown in **Figure 6.14**. Following experience gained from section 6.4.2, the first step to be taken in this scenario is to divide multiple outputs into several sub-systems each of which has the same multiple inputs but only one output. The general metamodel of each sub-case can then be adapted into equation 6.12. The output time for the sub-system can be treated as an array:

$$\begin{pmatrix} \beta_{01} + \beta_{11} WIP_1 + \beta_{21} T_{arrivA} + \beta_{31} T_{arrivB} + \varepsilon_1 \\ \cdot \\ \cdot \\ \beta_{0n} + \beta_{1n} WIP_n + \beta_{2n} T_{arrivA} + \beta_{3n} T_{arrivB} + \varepsilon_n \end{pmatrix} \qquad (6.14)$$

here $\beta_{2i}$ and $\beta_{3i}$ (i = 1, 2, .....n) should be near 1 as well.

The operation time in such metabox could be:

The operation time in such metabox could be:

$$\begin{pmatrix} \beta_{01} + \beta_{11} WIP_1 + \varepsilon_1 \\ \cdot \\ \cdot \\ \beta_{0n} + \beta_{1n} WIP_n + \varepsilon_n \end{pmatrix} \tag{6.15}$$

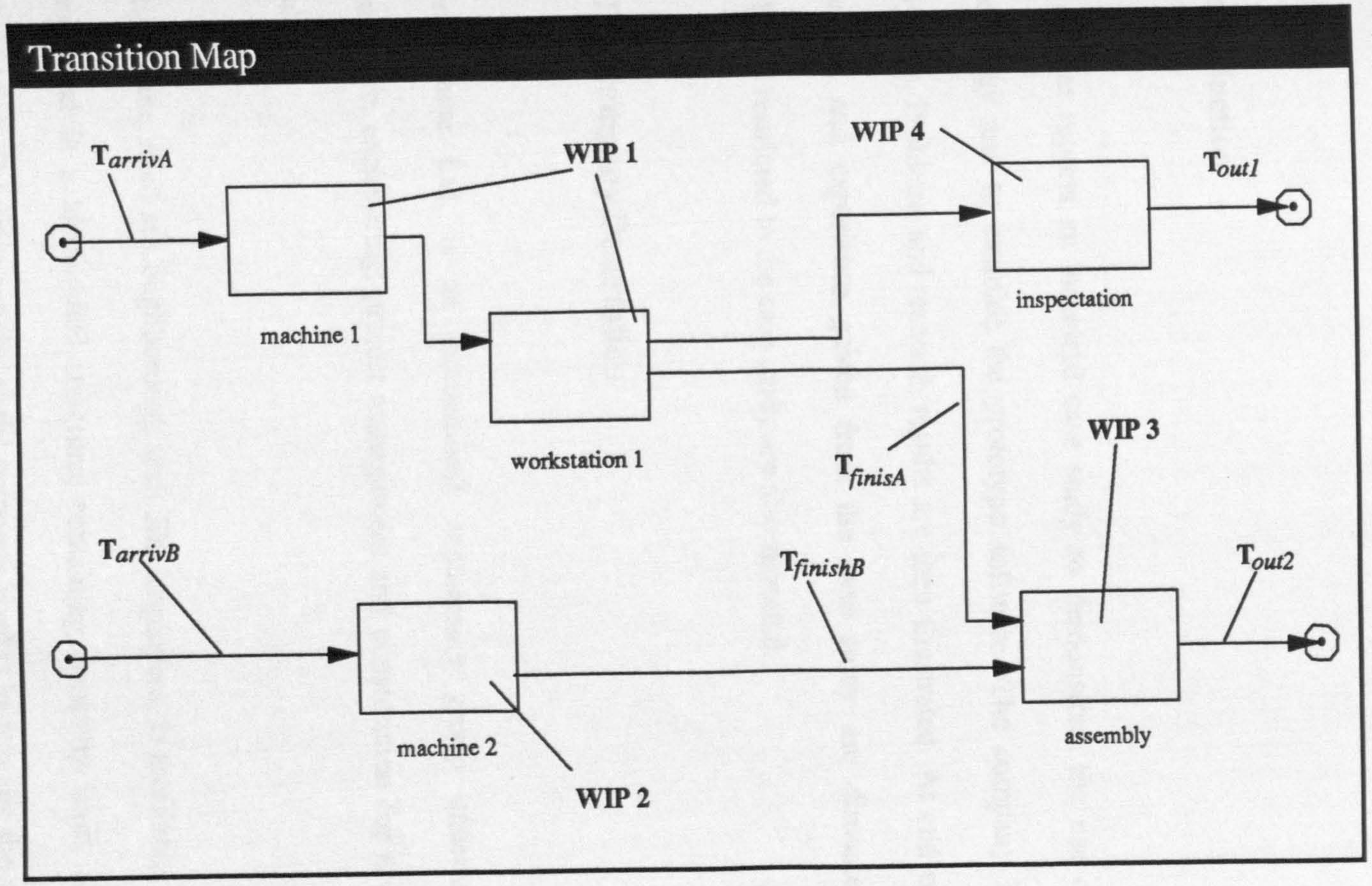



**Figure 6.14 Multiple Inputs and Multplei Outputs Simulation Scenario**

# Chapter 7

# CASE STUDY

## 7.1 Introduction

This chapter reports an industrial case study to demonstrate the use of the hybrid methodology and to examine the prototype software. The company is introduced briefly first. Problems and research results are then illustrated. At end of the chapter knowledge and experience gained from the case study are discussed and some conclusion restricted to the case study are also reported.

## 7.2 The Company Description

Morris Crane Ltd. is an international engineering group undertaking design, manufacture, engineering, project management and construction for a wide range of industries.

Morris Crane, sited at Loughborough with 300 employees, is profitable. The company is organised in a hierarchical structure, containing company level, shop level and machine level. The main activity at the company level is to manage the pre-production activities. The shop used in the company is similar to that of other companies, and machines carry out the real manufacturing operations on the shop floor.

### 7.2.1 Operation of the Company

As an independent trading unit, the company is managed under its own operating system. The company level consists of several functions, including sales and marketing, design, production control and engineering and purchasing. The domain of shop level is around manufacturing system which is organised in a cellular structure. Each cell is defined as the basic processing unit in the company.

The production of the company is mixed One-Of-a-Kind and Make-To-Stock, based on the type of product. Once a customer order is received by the sales and marketing section, a specification list will be passed on to the design office, whilst the estimator will provide approximate cost on an overall contract if it is not a standard product. The quote and approximate lead time is then fed back. When an order is confirmed, the contract begins to be processed.

The design section works on an order enquiry or a final contract order. The new design or modified design is made according to customer specification. When the drawings are finished, they are released to the production control and engineering section where a production schedule is produced and the drawings are examined by the production engineer, while a material requisition list is generated and sent to material control section.

The production control and engineering section has two functions. The first is to manage production in both long-term planning and the daily activity on the shop floor.

The second is to manage the material control section. It controls the raw material and sub-assembly parts released to the shop based on the drawings and the schedule. It informs the sales and marketing section of transport arrangements as well, when a finished product meets the customer's requirement and design specification.

The function of the material control section is clear, it includes purchasing material and parts, receiving goods, release of raw material and finished goods. It has a strong link with stock where raw material and sub-assemblies are stored and is located at production shop. The section has another function which is to control and store completed product.

Depending on the product (wire rope hoist, chain hoist and crane), the manufacturing activities are operated in two shops. One is for hoist production and the other is for crane production. The hoist shop comprises three cells, namely fabricate cell, chain hoist assembly cell and wire rope hoist assembly cell.

### 7.2.2 Products and Production Procedures

The company produces a wide range of lifting products, including heavy duty, standard industrial and special-purpose cranes, hand and electric chain hoists and wire rope hoists. Chain hoists and wire rope hoists are the main product of the company.

Two assembly lines produce all kinds of hoists. Although these products differ from each other, the major procedures for the each type of hoist remain the same. Figure 7.1 shows the process procedure of wire rope hoist.

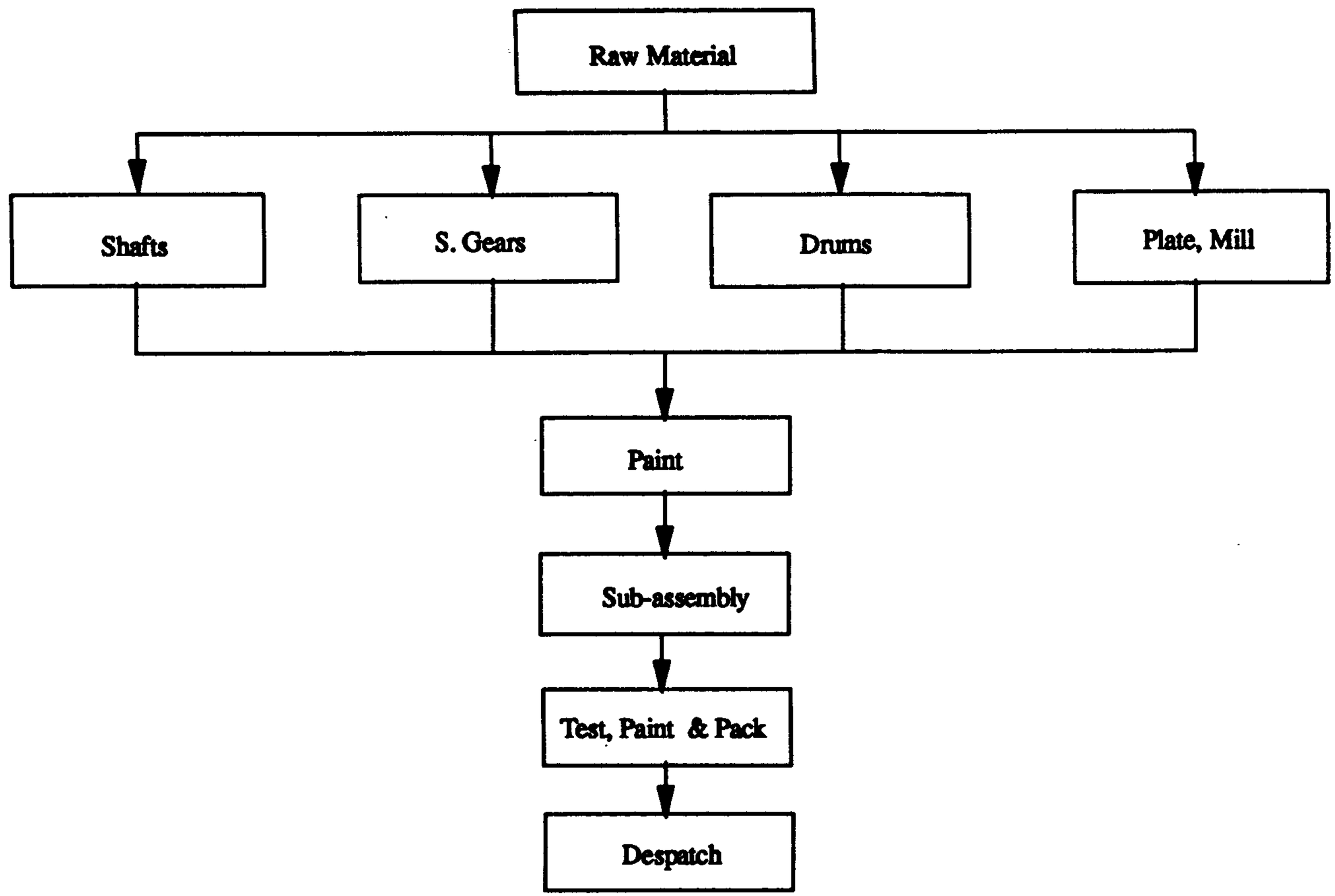



**Figure 7.1 Production of Wire Rope Hoist**

## 7.3 The Case Study

The case study had two objectives, these were: 1) to test the methodology established with realistic industrial data and operational strategies, 2) to analyse performance of the hoist division where competitive capability is willing to be improved.

### 7.3.1 Problem Tackled

In the division, producing a wire rope hoist takes longer than it should. This causes the company to lose its competitive capability in the market. So, the aim of the study is to get an insight into the behaviour of the production function and to identify possible ways of reducing the lead time. The problem identified was as follows:

- shortages are the major production problem because assembly is the biggest restriction to output growth and it is not easy to buy-in capacity as in the machine shop

- parts kitting in bulk batches with too many parts extends lead time and increases WIP. This causes typical lead times 3-4 weeks in which assembly activities takes only 3-4 days.

Therefore, effect was first focused on the assembly area. The metamodelling technique was applied in this area in order to view the behaviour in quantity. After the exploration of the production assembly area, the stores area was also investigated in

detail, for two reasons, these being a) the function is relevant to explore the material shortage problem, b) examining the hybrid modelling methodology in aid of large system simulation modelling design.

### 7.3.2 The Description of Production Function

The production functions in the division are divided into machine shop area, assembly area, painting area and testing area. The production supervisor is responsible for the shop control, loading and scheduling of jobs in these areas except for the loading and scheduling of jobs for the machine shop which is done by Production Control.

Once the Sales receives an order, the Production is announced. The job is then left until a week before the due date. The weekly assembly schedule is prepared when a "Wire Rope Hoist Loading Report" is received. This schedule is used to produce marshalling schedule and daily schedule. The marshalling schedule is sent weekly to the Store for Kitting. The daily schedule is used to issue "Drawings" and "Part Lists" to each team for each assembly job. The assembled basic units are then queued in a sub-assembly bay and wait to final customising assembly. After finishing the customising assembly, the product is moved into the finishing area for test, painting and finally packing.

## 7.4 Experiment Approach

Initially, a static model of the division was constructed in order to get essential

knowledge of the manufacturing operation system. After discussion with the manager and people who had knowledge of the system, the static model was modified. Then, building a dynamic simulation model started taking the static model into consideration. **Figure 7.2** shows the experimental stages and step-by-step procedure adopted in the study. The tools used for achieving the objective is displayed in **Table 7.1**.

| Purpose | Method / Technique | Tool |
|---|---|---|
| System description | Simulation | IDEM software |
| Metamodel construction | Regression analysis | Developed by the author |
| Metamodel validation | Double-cross | Developed by the author |

**Table 7.1 Adopted Tools for the Case Study**

### 7.4.1 Simulation Modelling

To carry out the case study, the simulation model was run under the different conditions which are specified by product type, input data and operation strategies. The model simulated manufacturing activities of three types of product, particularly in assembly area. The detailed product structure are displayed in **Table 7.2** to **Table 7.4**. **Table 7.5** shows an example of input information under random process. Other input information includes customer order information, inventory information and capacity information. These input values were provided by either accurate data or statistical data.

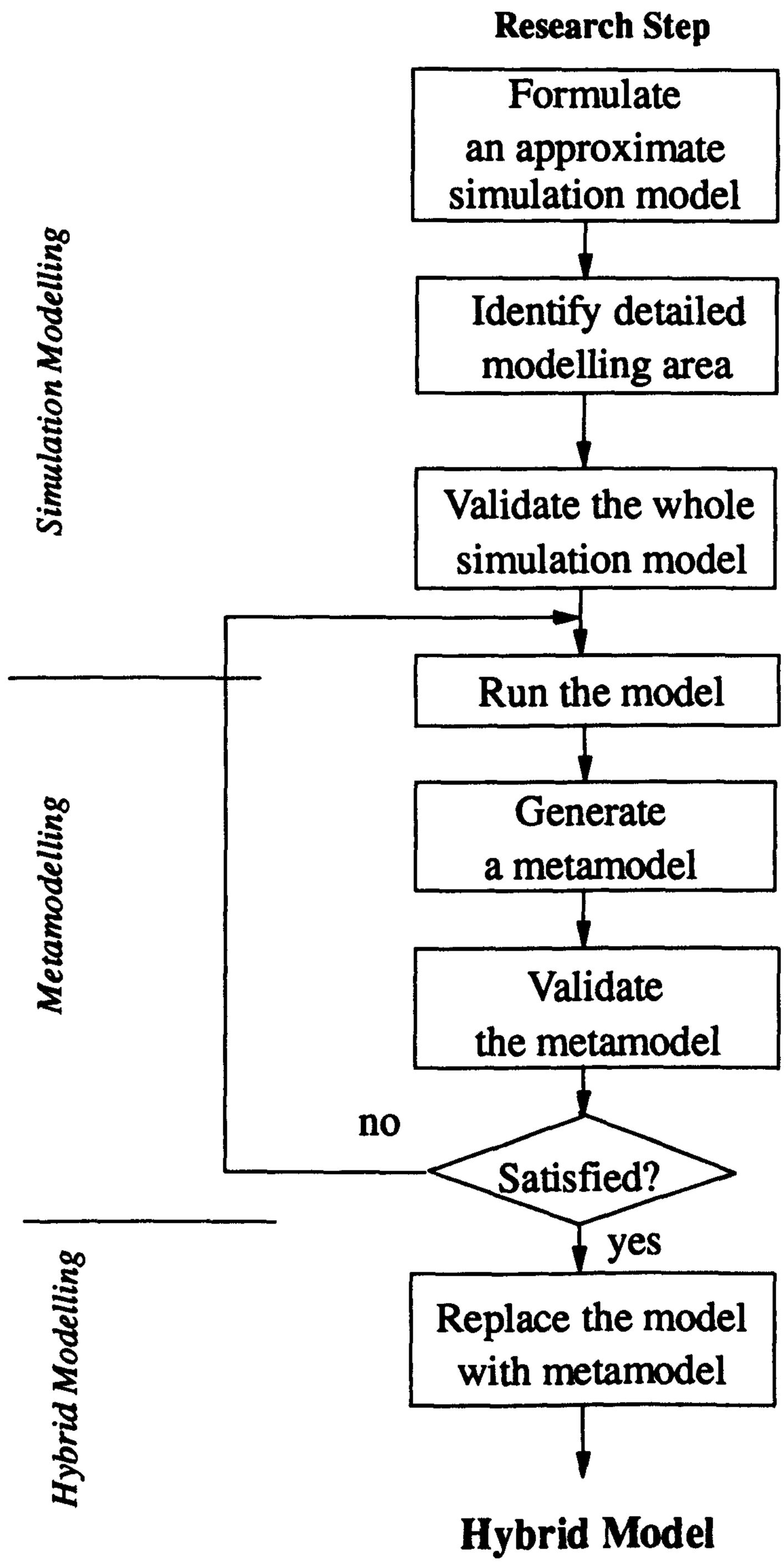



**Figure 7.2 Execution Plan of Case Study**

```
((A212  1  (NIL)
  A410  1  (NIL)
  A114  1  (NIL)
  A132  1  (A212)
  A210  2  (A410)
  A143  1  (NIL)
  A105  1  (NIL)
  A123  1  (A114 A143)
  A111  1  (A132 A105)
  A306  1  (A111)
  A214  1  (NIL)
 Order A  1  (A306 A 214 A210))
```

**Table 7.2 Componentsof Product A**

```
((B188  1  (NIL)
  B214  1  (NIL)
  B234  1  (NIL)
  B267  1  (NIL)
  B256  1  (NIL)
  B333  1  (NIL)
  B283  1  (NIL)
  B202  1  (B214 B267)
  B406  1  (B234)
  B294  1  (B202  B283 B188 B256)
  B211  1  (B333)
 OrderB  1 (B211 B294))
```

**Table 7.3 Componentsof Product B**

```
((C214 1 (NIL)
  C153 1 (NIL)
  C312 1 (NIL)
  C136 1 (NIL)
  C124 1 (C214)
  C132 1 (C124)
  C111 2 (NIL)
  C212 1 (C153)
  C118 1 (C312 C136)
  C233 1 (C132)
  C285 1 (NIL)
 OrderC 1 (C285 C111 C212))
```

**Table 7.4 Componentsof Product C**

| Input Rules | |
|---|---|
| Order Release Interval: | 10 (POSSION 10) |
| Assembly Daily Schedule Rule: | Duo Date |
| Product Release Selection: | To shipping |
| Product Inspection Rule: | Shipping out |
| Product Safety Stock: | 1 |

**Table 7.5 Input Information Under Random Process**

Operating strategy is another requirement for the simulation model. The operating rules include customer order releasing, material control, daily scheduling, inspection and inventory management. The rule of the daily schedule and assigning of fitters for each assembly job in queue is based on:

- priority according to due work week,
- priority given to commercial pressure.

The rule of material control is defined:

- re-order level depending on history data,
- release the sub-assembly parts for the high priority jobs.

In the detailed simulation level, independent variables have been identified in order to study the effect of independent variables on Lead Time - the dependent variable. The independent variables are Demand, Time for sub-assembly, Time for final-assembly. This led to a full factor experimental design being applied in the study. It is represented in **Table 7.6** where (+) represents the "upper" and (-) represents the "lower" values of the independent variables.

| Demand | Time for sub-assembly | Time for final assembly |
|---|---|---|
| + | + | + |
| + | + | - |
| + | - | + |
| + | - | - |
| - | + | + |
| - | + | - |

| | | |
|---|---|---|
| - | - | + |
| - | - | - |

**Table 7.6 Experimental Design**

The "high" and "low" values of the independent variables are displayed in **Table 7.7**.

| **Independent Variable** | Low Value (-) | High Value (+) |
|---|---|---|
| Demand | 1 | 4 |
| Time to sub-assembly | (6, 7) | (9, 10) |
| Time to final-assembly | (11, 13) | (17, 18) |

**Table 7.7 Data of Simulation Experiment in Detailed Level**

The running of the simulation model was conducted by the **Table 7.6**. Each experiment was repeated three times with different rules (means daily scheduling rule and material control rule) combination and other input data patterns.

### 7.4.2 Metamodelling

To generate a metamodel, both independent and dependent variables need to be identified. Considering the problem described above, initial condition data *Demand* is transfered into *batchSize* and *productType* form on detailed modelling level in this case.

The time for sub-assembly and the time for final assembly can be seen:

$$T_{sub-assembly} = f(WIP, batchSize, productType)$$
$$and$$
$$T_{final-assembly} = f(WIP, batchSize, productType)$$

Therefore, the independent variables were identified as:

Work-In-Progress,

Batch size, and

Product Type

whilst lead time is the dependent variable.

## 7.5 The Results of Case Study

### 7.5.1 Simulation Model Representation

A simulation model is created with IDEM software to investigate the exposed problem. **Figure 7.3** to **Figure 7.7** represent the approximate model which describes an overall structure of the company. Because only the assembly area and the store are of interest in detail, the rest of the simulation model is kept at an approximate level.

The detailed model of the assembly area and the store is illustrated in **Figure 7.8** and **Figure 7.9**, respectively. The output from the experiment is shown in **Table 7.8**.

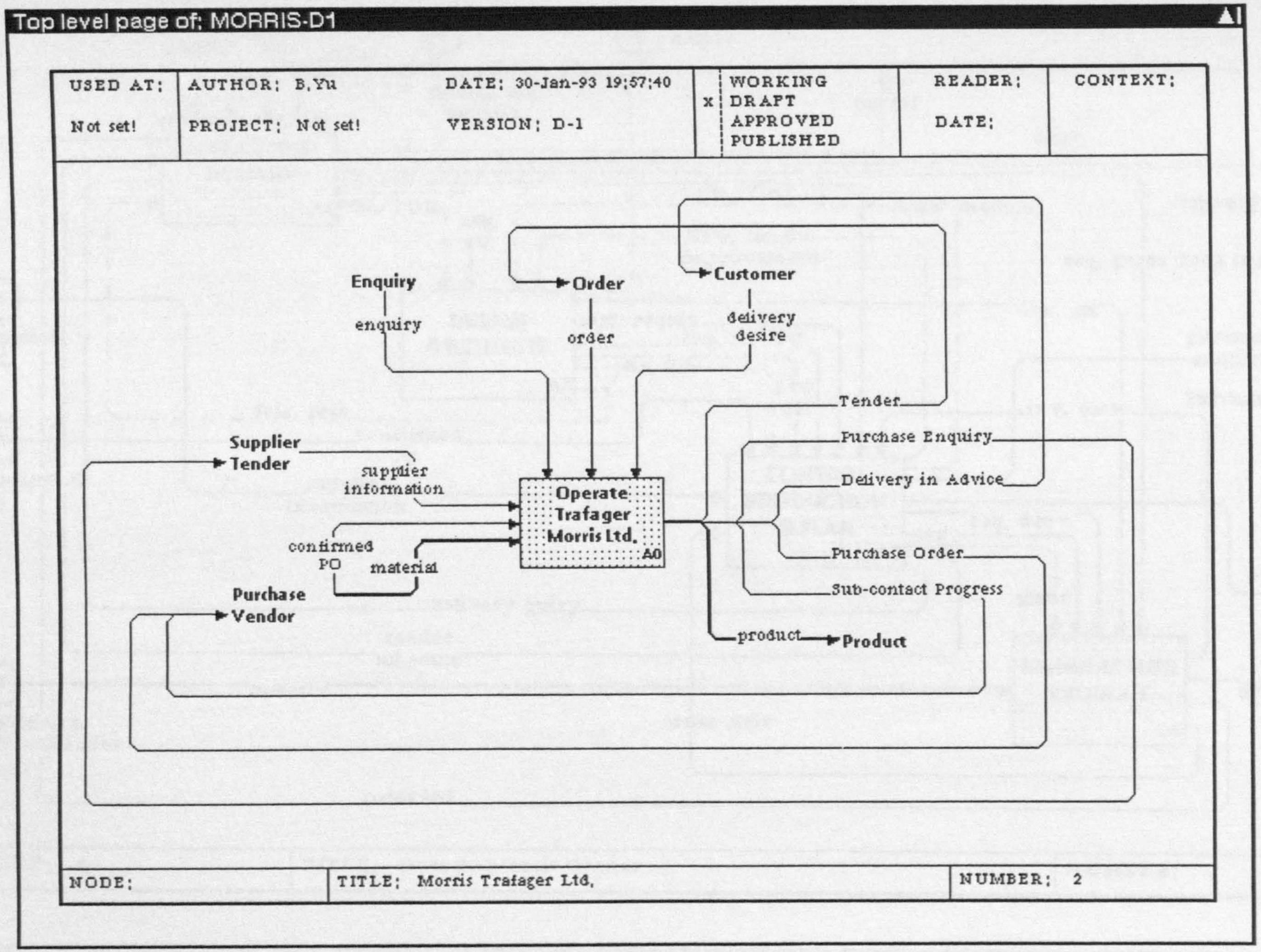



**Figure 7.3 The Top Level Page of the Simulation Model**

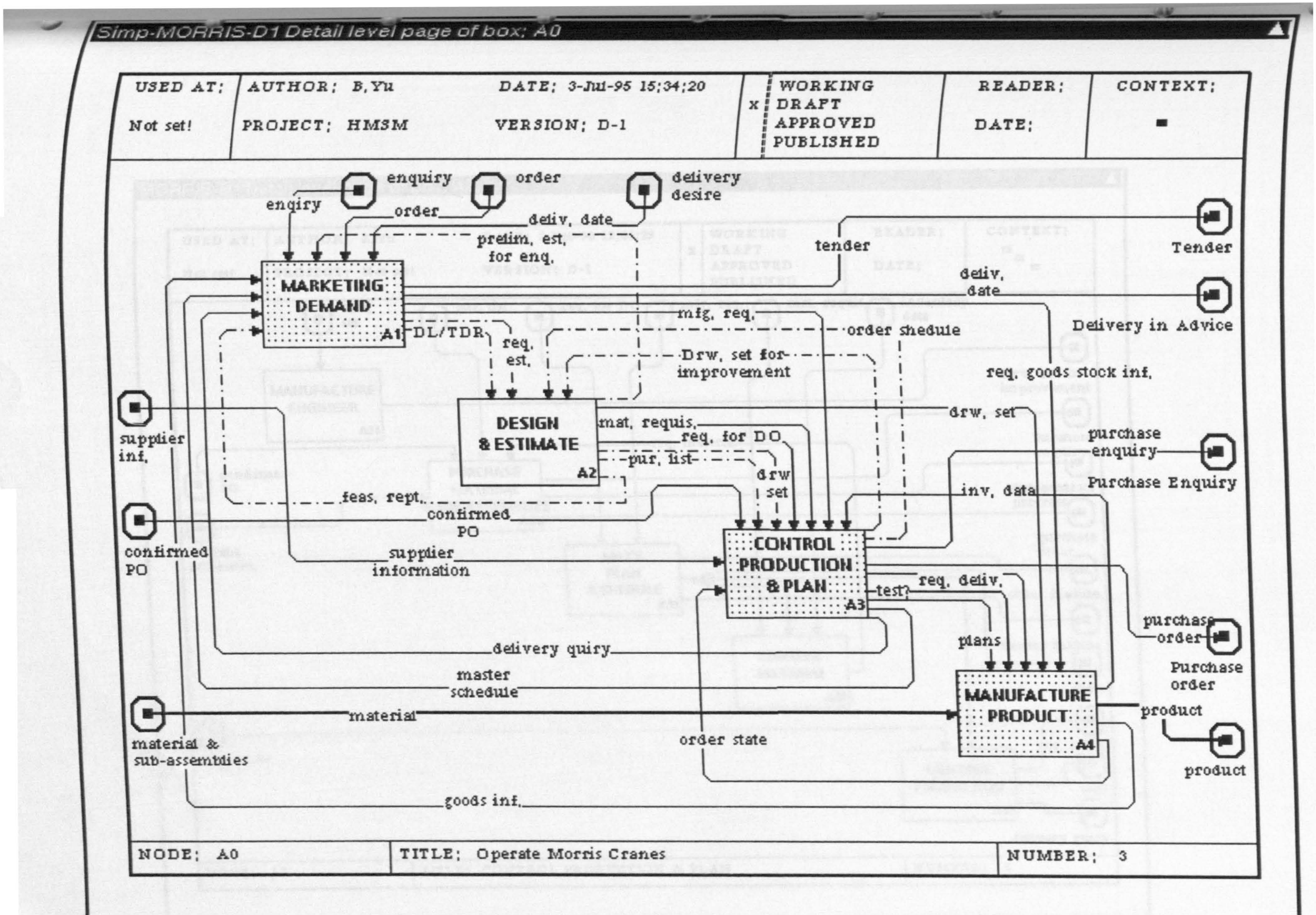



Figure 7.4 The Simulation Model at Approximate Level (1)

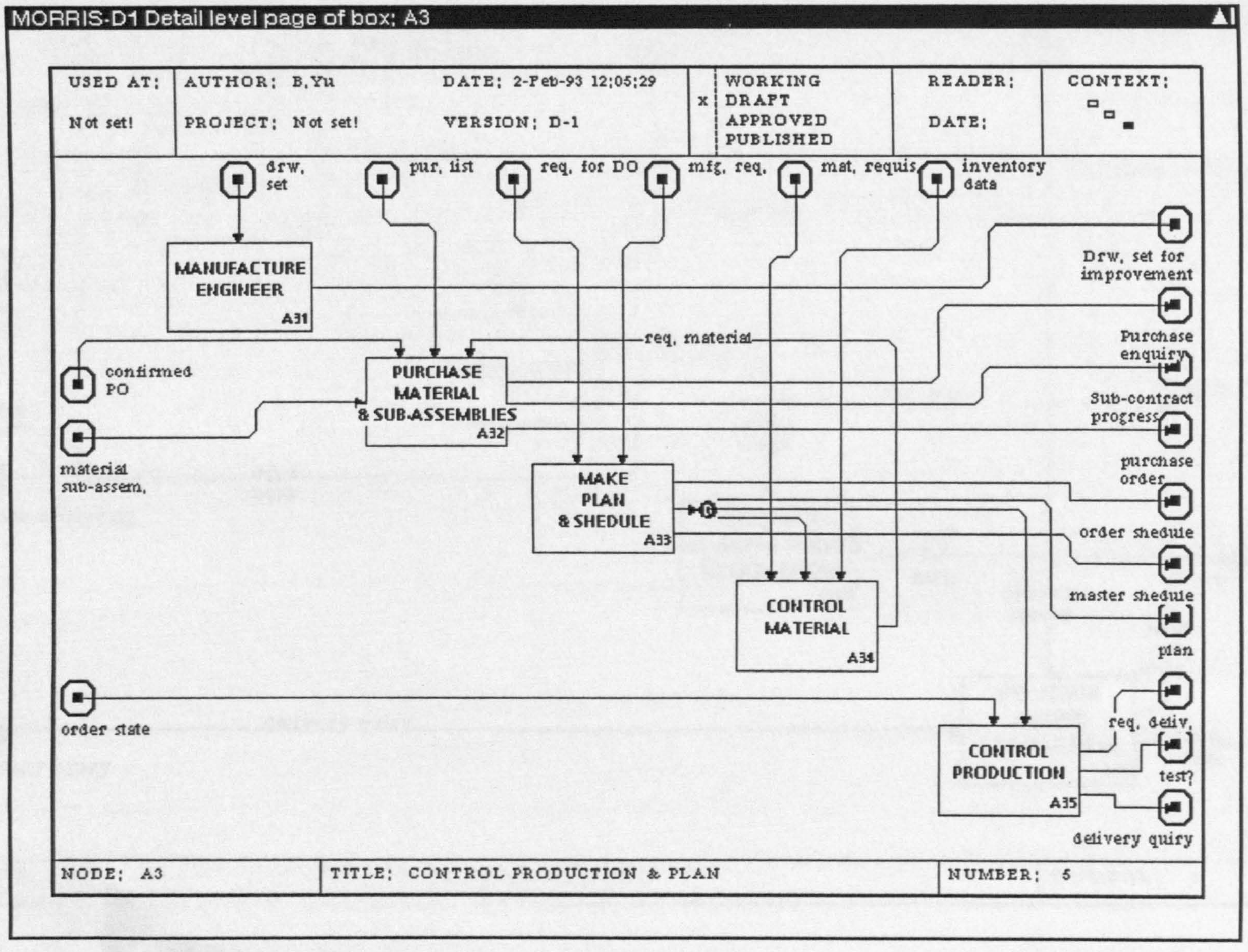



**Figure 7.5 The Simulation Model at Approximate Level (2)**

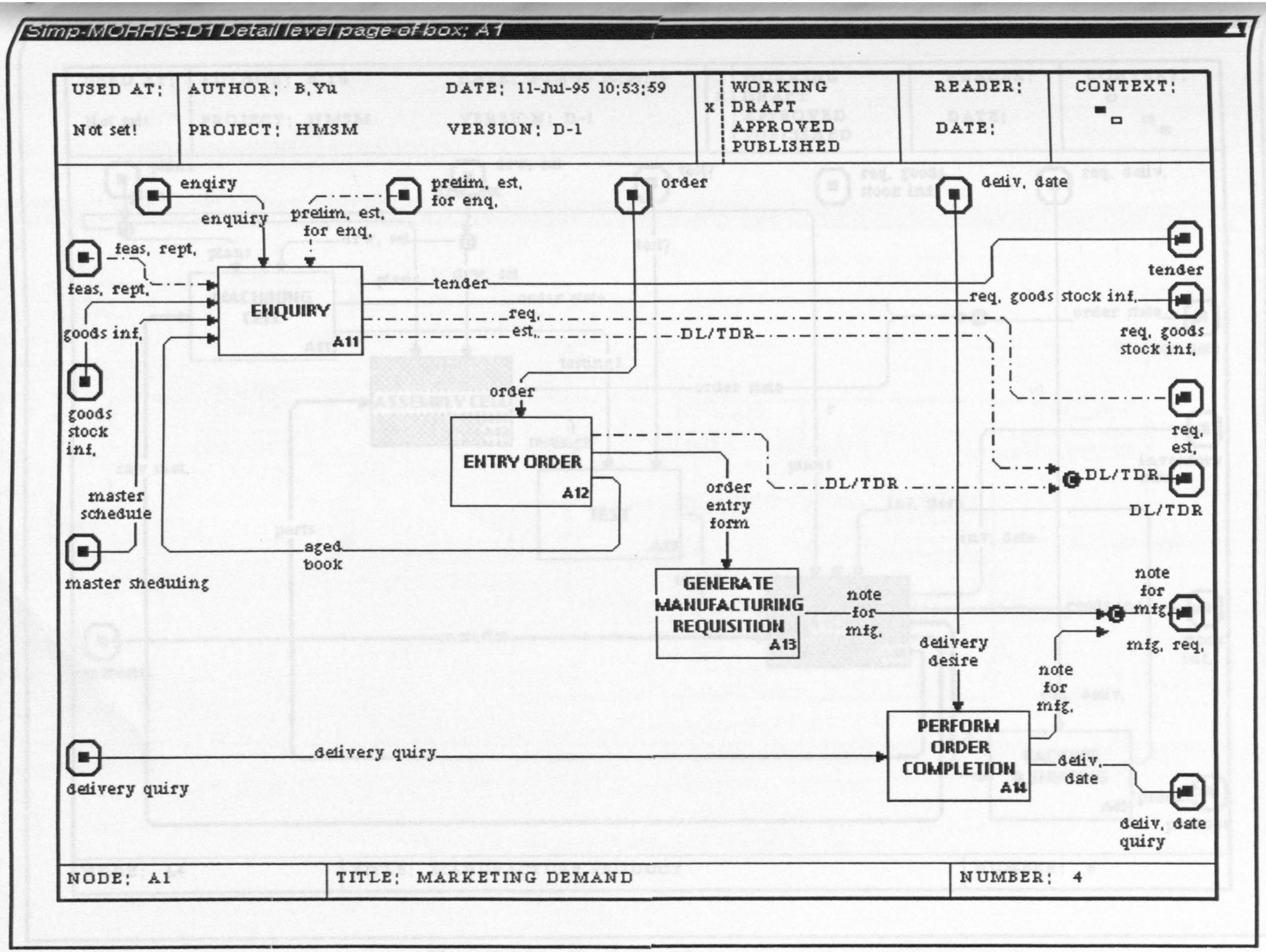



Figure 7.6 The Simulation Model at Approximate Leve (3)

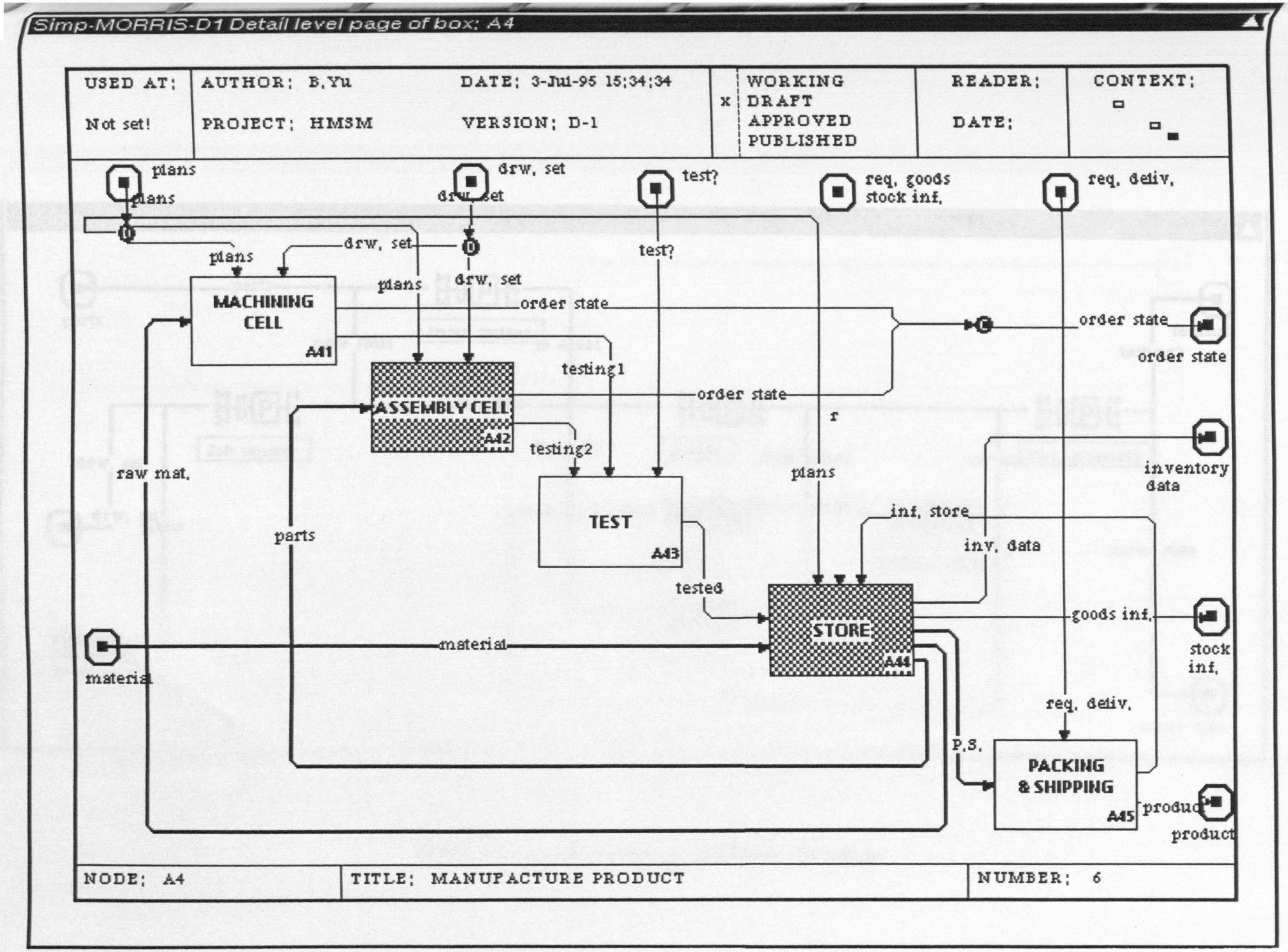



Figure 7.7 The Simulation Model at Approximate Leve (4)

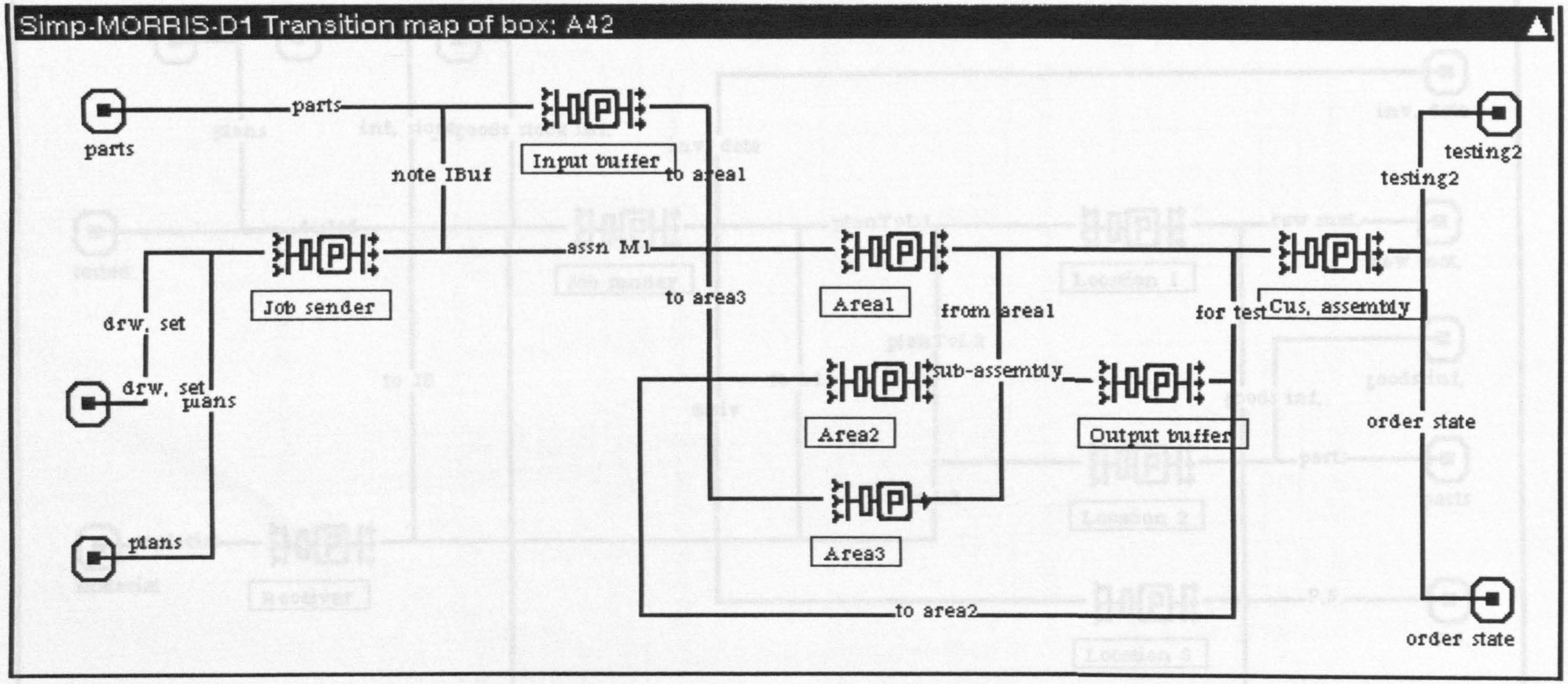



**Figure 7.8 Assembly Shop1 of MORRIS - D1 Model**

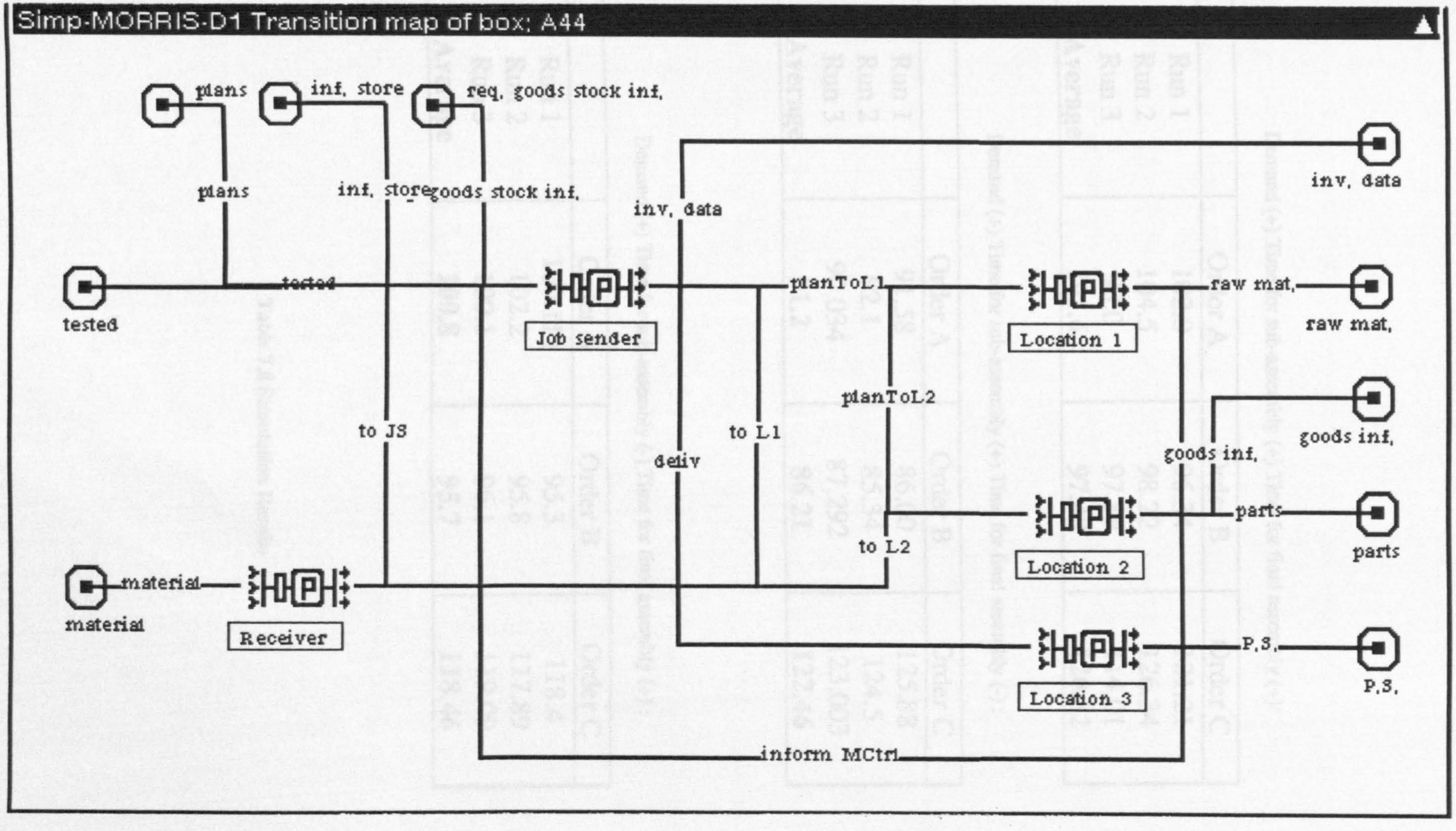



Figure 7.9 Stock of MORRIS - D1 Model

**Demand (+) Time for sub-assembly (+) Time for final assembly (+):**

| | Order A | Order B | Order C |
|---|---|---|---|
| Run 1 | 102.0 | 96.34 | 121.21 |
| Run 2 | 104.5 | 98.22 | 126.34 |
| Run 3 | 98.0 | 97.89 | 114.01 |
| **Average** | **101.6** | **97.48** | **120.52** |

**Demand (+) Time for sub-assembly (+) Time for final assembly (-) :**

| | Order A | Order B | Order C |
|---|---|---|---|
| Run 1 | 91.58 | 86.00 | 125.88 |
| Run 2 | 92.1 | 85.34 | 124.5 |
| Run 3 | 90.034 | 87.292 | 123.003 |
| **Average** | **91.2** | **86.21** | **122.46** |

**Demand (+) Time for sub-assembly (-) Time for final assembly (+) :**

| | Order A | Order B | Order C |
|---|---|---|---|
| Run 1 | 100.111 | 95.3 | 118.4 |
| Run 2 | 102.2 | 95.8 | 117.89 |
| Run 3 | 100.1 | 96.1 | 119.09 |
| **Average** | **100.8** | **95.7** | **118.46** |

**Table 7.8 Simulation Results**

### 7.5.2 Development of Metamodel

Figure 7.6 shows this detailed sub-system is a multiple inputs and one output case. Before developing a metamodel, simulation results of the dependent variable and the independent variables must be available. Obviously, the value of the independent variables identified in section 7.4.2 can be captured directly, but the dependent variable cannot be gained directly. It is therefore necessary to analyse how to get the values of the dependent variable.

Logically, the assembly activities start when the schedule arrives at the shop, assuming the drawings should be sent with the schedule, and materials/parts will be received soon after the schedule arrival. Therefore, the scenario can be simplified as one input and one output case. The lead time is $T_{output-release} - T_{schedule-arrival}$. Both output release time and schedule arrival time can be captured from the simulation model directly. If there is a shortage of materials or parts, the lead time is extended because output release time is delayed.

After a set of simulation experiments, which is conducted by a experimental design table shown in **Table 7.6**, the metamodel is formulated and illustrated as below:

$$LT = 15.2 + 8.02WIP + 3.6B + 3.03P + 0.22B \times P \qquad (7.1)$$

### 7.5.3 Validation of Simulation Model and Metamodel

To validate the simulation model, the company's manager was consulted and the real data was compared with the data generated by the simulation model.

At this stage, different input data combinations were selected and entered into the simulation model, this resulted in a great deal of simulation outputs. Through the communication with the company manager, who has the knowledge of the system. The simulation model was then modified in both structure and information. Eventually the simulation model was validated.

As described in chapter 5, a user friendly validation function has been included in the experimental software. Therefore, selecting *Validation Metamodel* from the menu accomplished the purpose. The validation results are displayed as follows:

| | Sample I / Coeff | Sample II/Coeff |
|---|---|---|
| Intercept | 15.1 | 15.06 |
| WIP | 8.0 | 8.1 |
| Batch size | 3.55 | 3.6 |
| Product type | 3.0 | 3.02 |
| R-sq | | |
| Selected | 0.799 | 0.80 |
| Unselected | 0.81 | 0.78 |

**Table 7.9 Validation Results**

From the above Table , two $R^2$s are similar. It implies that the metamodels are valid.

Although the regression model was highly significant (F-ratio = 7.9 and p= 0.001) and a reasonably high value of $R^2$ = 80 was obtained, $s^2$ value was 0.564, which was considered high. To ensure the prediction was reliable from the model, the model assumption, namely constant variance checking, was carried out next.

A constant variance check was made by regressing the square of residuals on 3 variables. $SS_{reg}$ for this was found to be 8.19. The following calculation was carried out:

$$S = \frac{1}{2} SS_{reg} \left(\frac{n}{(n-p)s^2}\right)^2 \qquad (7.2)$$

where n = 96 (number of observation)

p = 3 (degrees of freedom)

$s^2$ = 0.564 (MS residual)

$SS_{reg}$ = 8.19 (SSreg of residual square).

This value was checked against the $\chi^2$ value at 95 per cent confidence level.

$$\chi^2_{0.05,4} = 14.86$$

As S < $\chi^2$, the null hypothesis which states that the variance is constant, was accepted. Therefore, the metamodel is reliable if it is used as a predictive model.

### 7.5.4 Hybrid Model vs. Simulation Model

The following step in the case study was to replace the detailed simulation model with its corresponding metamodel, expressed in **Equation (7.1)**. The replacement was accomplished through the menu which was written as part of the research. When the metamodel was inserted into the simulation model, the hybrid model was obtained and tested. The results of running the hybrid model are shown in **Table 7.10**. Comparing the results obtained through the simulation model, see **Table 7.8,** the average deviation was found to be within 1.79 per cent, shown in **Table 7.11**. Therefore, it proves that the hybrid model can predict equivalent results to those predicted by the simulation model.

In order to test the methodology as an aid for large system design, another box has been selected for detailed investigation after the replacement. This detailed simulation model is illustrated in **Figure 7.9**. The experiment was carried out through two models with the same input pattern. One model is the simulation model in which the manufacturing box was kept as the detailed simulation model. The other was the hybrid model in which the manufacturing box was replaced by its corresponding metamodel. The average time spent in generating a simulation output was observed. If a large simulation model is used, obviously, the hybrid model can save significant computing times.

## 7.6 Case Study Experience

Through this case study, the author gained certain knowledge and experience. They are not only related to the manufacturing fields but also to the simulation techniques. They are:

- Priority given by order can only satisfy the particular order, and it causes more delay to the other orders. Therefore, daily schedules should be based on kits with all parts available for assembly.

- Do not start partial assembly if there is a shortage of parts to complete a assembly job. Only if there is not any order with complete materials available for assembly.

- Stores should advise production control if there is a materials/parts shortage. By so doing, the daily schedule can be changed in order to save the assembly capacity and to avoid increasing WIP.

- Different rules can greatly influence the system performance. For instance, the scheduling rules at the assembly area can effect the performance of finishing time.

- Approximate modelling can help to build the simulation model and analyze the system's overall performance quickly. This is useful for system design, particularly at an early design stage. This can lead to identification of further areas for investigation by detailed modelling.

- The case study experience has shown that the management of the data base in the model at the approximate level is efficient. Since detailed modelling requires more data than that of in approximate modelling level, some data has to be modified in the model. Therefore, the study has indicated that the modification utility needs to be improved to assist users who have less knowledge of LISP and LOOPS to change data easily.

## 7.6 Conclusion

This chapter has described the use of the prototype hybrid modelling software to simulate a large system. The case study has provided an opportunity to design a large system using the hybrid simulation technique. In particular,

- It demonstrates that the hybrid methodology has distinct advantages over conventional system design methodology. The computation time consumed has decreased when the hybrid model was running.

- The model indicates that the information system of the company, especially in material control, needs to be modified in order to overcome the material shortage problem.

| Demand | Time for Sub-assembly | Time for Final Assembly | Lead Time |
|---|---|---|---|
| + | + | + | 99.6 |
| + | + | - | 92.0 |
| + | - | + | 101.28 |
| + | - | - | 81.2 |
| - | + | + | 26.88 |
| - | + | - | 23.0 |
| - | - | + | 23.9 |
| - | - | - | 23.79 |

**Table 7.10 Results of Hybrid Model**

| Simulation | Hybrid Model | Deviation (per cent) |
|---|---|---|
| 101.6 | 99.6 | 3.2 |
| 91.2 | 92.0 | 1.35 |
| 100.8 | 101.28 | 0.7 |
| 79.6 | 81.2 | 2.68 |
| 26.76 | 26.88 | 0.8 |
| 22.7 | 23.0 | 2.13 |
| 24.28 | 23.9 | 2.66 |
| 23.9 | 23.79 | 0.79 |
| | | ***Average:*** 1.79 |

**Table 7.11 Hybrid Model Results Vs Simulation Model Results**

# Chapter 8

# EVALUATION AND DISCUSSION

## 8.1 Introduction

Chapter 7 elicited some results on the major features of the research work based on the case study. This chapter will illustrate the achievement of the research itself. In the meantime, it also identifies the limitation of the research, provides some clues to requirements for the further developments and points out the potential directions of future research works. Hence, this chapter aims to appraise the research work exhibited in this thesis critically.

## 8.2 The Achievements of the Research

From the descriptions of the research principle and research implementation, and from the demonstration of the metamodel application, this research has succeeded in making significant progress on modelling a larger system with less computer time and cost. The major achievements are:

- It proposes and demonstrates a new system design methodology, called the hybrid methodology, which integrates the capability of simulation modelling and mathematics modelling. It accomplishes a desire to model a system with simulation technique and mathematics in a single environment. With this

method, behaviour of a larger system can be examined in more detailed areas.

- The research identifies that the hybrid method has achieved in reducing the requirement of computer time and cost. Although model validation is a very important stage, it has always represented a serious time-consuming problem, particularly when a large model needs to be validated. This research has contributed to improve the situation.

- Metamodel, which is constructed here with the regression analysis method, can be trusted. The results of the case study certify that a metamodel can be used to generate reliable simulation results both in quantity and quality.

## 8.3 The Limitation of the Research

Since certain conditions and assumptions have been made, this research is subject to the following limitations:

1. The assumptions of simulation model

The assumption of this study is that the simulations are at steady-state. Transient behaviour of a simulated system must be omitted so that regression analysis can be used. If the transient state is included, it is too difficult to find a solution or may be impossible to have one by using regression analysis. In this case, other methods for metamodelling construction may be required.

2. The software and hardware constraints

The software limitation is the constraints of the prototype software development environment - LISP and IDEM. LISP is an advanced programming language in knowledge representation, yet it has less functionality in algebra. This causes more difficulties in developing an efficient metamodel generator. Although there are many existing statistical packages available, they can not be run under LISP environment.

However, it is necessary to point out that this limitation has no effect on the principle of the hybrid methodology, because the principle is based on the basic simulation principle and metamodel concept (refer to chapter 5). Since the prototype software is developed in a LISP environment, the limitation does have the sole impact on the demonstration software.

The other disadvantage is the limitation of computing power. The available computer facilities cannot run a complicated model with a large amount of processes efficiently. In other words, it is easy to model a system which contains similar processes or sub-processes. This makes a restriction on the IDEM software itself and narrows down the exploration boundary of a simulation model. Therefore, these software and hardware constraints have reduced the accuracy level of the metamodel.

## 8.4 Discussion

In terms of factory design, it involves many different aspects such as time attributes on shop control, human resources in office design, impact of material shortages in shop floor, purchasing cycle and so on. However, the regression analysis can only identify the most critical independent variables which affect the response in steady state. It is not suitable when the system is in transient behaviour state, such as the impact of machine broken down on shop floor.

Clearly, one of the further study directions is to research on constructing metamodels with time series. The reason is that time is a vital independent variable in any manufacturing system. The method used to formulate a metamodel should be able to reveal time attribute explicitly, and time series method can meet the requirement.

Due to the time constraints, many other valuable areas have not been explored. They are:

1) impact of metamodels generated through other methods

2) interpretation facility, and

3) delivery of expert systems to enrich the usage of metamodel

In the current research, the metamodel generator works according to the principle of regression analysis method. Thus, there is need for research on the impact of metamodel generation with other methods such as piece wise.

Because embedding a metamodel in simulation is the central issue in this study, further explorations of the metamodel usage are beyond the research objectives. However, the metamodel can be enriched in many areas, such as adding an interpretation facility which can suggest further simulation investigations in order to justify the simulation model. Although many authors have concluded that metamodels can play a role guidance to conduct further simulation studies, a comprehensive system has not been found to help the user for this purpose. Since the metamodel has already been formed, it is worthwhile and possible to develop a good interpretation facility to assist the researcher for this purpose.

One of the aims of user interface is to ease metamodel validation. Although the program written for the research can validate the metamodel, it still requires a user to have comprehensive statistical knowledge to verdict their model's validity. Since the supporting theories are mature, the metamodel validation utility should be customerized further to assist users who have less statistical knowledge to validate their metamodel.

Another aim is to re-use those valid metamodels. Since several sub-systems of a larger system can have similar or even the same processes, a metamodel for the sub-system only needs to be generated once and then applied several times. It is not necessary to run the same simulation model of a sub-system more than once for obtaining the same metamodel.

One of the major benefits of using simulation is that it takes into account the dynamic

behaviour of the system studied. Several authors have considered that this dynamic behaviour is a particular type of knowledge (O'Keefe, 1986). Normally, description of the knowledge in simulation is in "what-if" form. However, this form cannot deal with these cases in which a situation is between two conditions, and it takes more time to get the right knowledge if the knowledge is at end of a large knowledge base. As we know that a metamodel is another kind of knowledge representation, it is possible to save a valid metamodel and re-use it when appropriate.

# Chapter 9

# CONCLUSION AND FURTHER WORK

## 9.1 Introduction

This chapter summarises the research work. It discusses the contributions related to the use and enhancement of the hybrid approach to design larger systems. It also recommends further studies in which research can be carried out to improve implementation based on the achievement of this research and to explore in design a new system with dynamic events.

## 9.2 Original Contribution

**The originality of the work is embedding a metamodel in simulation so that a large system can be modelled. The research work has proved that the hybrid methodology is capable of facilitating the delivery of a new system design approach, which possesses both the advantages of simulation and of mathematical modelling.**

## 9.3 Summary of the Research Results

Having achieved the objectives set at in section 2.3, the detailed contributions of this research are:

- The research has proven, via the development of the embedding metamodel, that it is a feasible practice to encapsulate the functionality of a mathematical model within a discrete event simulation environment.

- The research has produced a metamodel generator as a partial solution, which needs to be addressed in the implementation of the consistency of modelling. The contributions also include: a) a data format to be used either in metamodel generation or in simulation reporting, b) a user-friendly interface.

- In meeting the project requirements in the IDEM methodology, the research has fulfilled a bottom-up approach for system design. This work has achieved the project targets as a whole and suggests a systematic bottom-up approach for system design.

- The research has also contributed towards the advancement of the IDEM tool in terms of enhancing the function of the IDEM node. The improvement of the IDEM tool achieved by this work is not only in zoom-in paradigm level but also in approximate modelling level.

## 9.4 Further Work

The following sets out further research work required along the hybrid modelling approach:

- One direction is to formulate metamodels by the time series method, so that the metamodel formulated either from regression analysis or from time series can be applied into a system performance according to its purpose.

- Because there are certain statistical packages available which possess more functions and diagram facilities, it should be noted that it is worthwhile to adopt them rather than to spend time on doing the same work. Therefore, it is clear that the second direction is to develop a cross-bridge program by which the user can use other existing statistical packages, such as SAS, StaticGraph, straight away.

- As the metamodel has been obtained, acceptance of the model will be determined by lack-of-fit test. An interpretation utility with a data-base and knowledge-base should be developed to help the user to determine the accuracy of the model and the necessity of creating a new model.

- Research into response surface method, and apply it into the system design to yield optimal results. As knowledge obtained can be saved into the knowledge base, the model can then be restructured and run under different explorations.

## 9.5 Conclusion

From the research results of the embedded metamodel in simulation environment, it is clear that this study has offered a new approach in system design and demonstrated its

potential value. Traditionally, it has been difficult to design a larger system, mainly because of the lack of an appropriate methodology and disadvantages of simulation. The research work has contributed significantly to the solution of those difficulties. In addition, a prototype hybrid modelling tool has been developed. The tool demonstrates the methodology by which embedding a metamodel in the simulation environment can be accomplished and is a new powerful way in system design.

# REFERENCE

Anon, 1985, *Medley Reference Manual Volume I: Language*. Venue.

Anon, 1985, *Medley Reference Manual Volume II: Environment Reference*. Venue.

Anon, 1985, *Medley Reference Manual Volume III: I/O Reference*. Venue.

Appel, S., Ayel, J. and Laurent, J.P. et al., 1989, Controller Module for Coordination of Supervision Activities on Production Management Systems. *Computer Integrated Manufacturing (Ed. Halatsis, C. and Torres, J.), Proc. of the 5th CIM Europe Conference*, 17-19, May, 1989, Athens, Greece, pp. 3-14.

Baines, R.W. and Colquhoun, G.J., 1990, $IDEF_0$ an Integration and Design Tool for Engineers. *Assembly Automation*, Vol.10, No.3, pp. 141-145.

Banks, J. and Carson II, J.S., 1986, Introduction to Discrete-event Simulation. *1986 Winter Simulation Conference Proceedings (Eds. Wilson, J.R., Henriksen, J.O. and Robert, S.D.)*, Dec. 8-10, Washington D.C., U.S.A., pp. 17-23.

Beck, K.N., 1984, Validating Regression Procedures with New Data, *Technometrics*, Vol.26, No.4, pp. 331-338.

Beerel, A.C., 1987, *Expert System: Strategic Implications and Applications*. Ellis Horwood Limited, pp. 10-85.

Begg, L. M. and Worsley, C. P., 1989, AUDITION--an intelligent simulation environment, *Simulation and AI*, Proc. of the SCS Western Multiconference, San Diego, California, Simulation Series, Vol.20 No.3 July, pp.71-76.

Ben-Arieh, D., 1986a, Knowledge-based Control System for Automated Production and Assembly. *Modelling and Design of Flexible Manufacturing Systems* (Eds. Kusiak, A.), Elsevier Science Publishers B.V., Amsterdam, pp. 347-365.

Ben-Arieh, D., 1986b, A Knowledge-based System for Simulation and Control of FMS, *2nd Int. Conf. on Simualtion in Manufacturing*, IFS Ltd., pp. 13-21.

Blanning, R.W., 1975, The Construction and Implementation of Metamodels. *Simulation*, June, pp.177-184.

Bitran, G.R. and Chang, L., 1987, A Mathematical Progamming Approach to a Deterministic Kanban System. *Management Science*, Vol.33, pp. 427-441.

Booch, G. 1991, Object Oriented Analysis and Design With Applications. Second Edition, *The Benjamin/Cummings Publishing Company, Inc.*

Bravoco, R.R. and Yadav, B., 1985a, A Methodology to Model the Functional Structure of an Organization. *Computers in Industry*, Vol.6 pp. 345-361.

Bravoco, R.R. and Yadav, B., 1985b, A Methodology to Model The Information Structure of an Organization. *Journal of Systems and Softwares*, Vol. 5, pp. 59-71.

Bravoco, R.R. and Yadav, B., 1985c, Requirement Definition Architecture - An Overview. *Computers in Industry*, Vol.6, pp. 237-251.

Brillinger, D.R. and Krishnaiah, P.R., 1983, Time Series in the Frequency Domain. *Handbook of Statistics*, no.3. North Holland, Amsterdam.

Buzacott, J.A., 1985, Modelling Manufacturing System. *Robotics & Computer-Integrated Manufactuirng* 2(1), pp. 25-32

Carrie, A., 1988, *Simulation of Manufacturing Systems*. Wiley.

Champeaux, D.D. and Fanrc, P., 1992, A Comparative Study of Object-oriented Analysis Methods. *Journal of Object-Oriented Programming*, March/April, pp. 21-33.

Chaharbaghi, K., 1990, Using Simulation to Solve Design and Operational Problems. *Int. J. Operational and Production Management*, Vol.10 No.9, pp. 89-105.

Chaharbaghi, K., 1991, DSSL II: A powerful Tool for Modelling and Analysing Complex Systems. *Int. J. Operational and Production Management*, Vol.11 No.4, pp. 44-88.

Chen, P.H. and Talavage, J., 1982, Production Decision Support System for Computerized Manufacturing Systems. *J. Manufacturing Systems*, Vol.1 No.2, pp. 157-167.

Cochran, J.K., 1988, Techniques for Asscertaining the Validity of Large-scale Production Simulation Models. *Int. J. Prod. Res.*. Vol.25 No.2, pp. 233-244.

Cochran, J.K. and Chang, J.C., 1989, Optimization of Multivariate Simulation Output Models Using a Group Screening Method. *Computers & Inductrial Engineering*. Vol. No. ,

Colquhoun, G.J., Gamble, J.D. and Baines, R.W., 1989, The Use of $IDEF_0$ to Link Design and Manufactuirng in a CIM Environment. *Int. J. Operation and Production Management*, Vol.9 No.4, pp. 48-65.

Colquhoun, G.J., Baines, R.W. and Crossley, R., 1993, A State of the Art Review of $IDEF_0$. *Int. J. Computer Integrated Manufacturing*, Vol.6 No.4, pp. 252-264.

Cook, H.E., 1992, Organizing Manufacturing Enterprise for Customer Satisfaction. *Manufacturing System* (Ed.) by Heim, J.A. and Compton, W.D., National Academy Press, pp. 116-127.

Delaney, W. and E. Vaccari, 1989, *Dynamic Models and Discrete Event Simulation.* New York: Marcel Dekker.

Doumeingts, G., Darricau, D. and Berard, C. et al., 1987, Design of Advanced Manufacturing Systems Using AI Techniques. *Proc. IXth ICPR*, Aug. 17-20, Cincinnati, Ohio, U.S.A., pp. 1-7.

Felix, T.S. Chan and Smith, A.M., 1992, Simulation Aids JIT Assembly Line Manufacturing: A Case Study. *Int. J. Operations and Production Management*, Vol.13 No.4, pp. 50-74.

Friedman, L.W., 1984, Establishing Functional Relationships in Multiple Response Simulation: The Multivariate General Linear Metamodel. *Proceedings of the 1984 Winter Simulation Conference*, Scoiety of Computer Simulation, pp. 285-289.

Friedman, L.W., 1986, The Analysis of Multiple Response Simulation Output Data: Experiments of Comparison. *Computer & Operations Research*, Vol.13, No.6, pp.647-652.

Friedman, L.W. and Friedman, H.H., 1984, Title. *J. of Statistical Computer Simulation*, Vol.19, No. 3, pp.237-263.

Friedman, L.W. and Friedman, H.H., 1985, Validating the Simulation Metamodel: Some Practical Approaches. *Simulation*,Vol.45 No.3, pp.144-146.

Geffrion, A.M., 1976, The Purpose of Mathematical Programming Is Insight, Not Numbers. *Interface*, Vol.7, No. 11, pp.81-92.

Goddi, P. and Keuneke, A. M., 198*, Medley Lisp Development Environment. *IEEE Computer Society*. August.

Goh, A.S., 1994, Step Toward the Design and Control of a New Generation of Machines. PhD thesis, Loughborogh University of Technology.

Goldstein, G.B. and Dushane, T., 1976, Repro-modelling Applied to the Simplification of Taradcom Computer Models. *Report No. 12243, US Army Tank/Automotive R&D Command, USA.*

Goyal, S.K., Deshmukh, S.G. and Babu A.S., 1991, Analysis of Integrated Procurement - Production Systems Using Mathemathical And Simulation Modelling Approaches. *Production Planning & Control*, Vol.2, No.3, pp. 257-264.

Hannan, E.J. Krishnaiah, P.R. and Rao, M.M., 1985, Time Series in the time Domain. *Handbook of Statistics*, no.5. North Holland, Amsterdam.

Hanson, W. C., 1992, The Integrated Enterprise. *Manufacturing Systems*, Ed. by Heim, J.A. and Compton, W.D., National Academy Press, Washington, D.C., pp.158-165.

Hatley, D.J. and Pirbhai, I.A., 1987, *Strategies for Real-time System Specification.* Dorset House.

Hicks, C., Braiden, P.M. and Simmons, J.E.L., An Assessment of the IDEF0 Methodologies and Their Suitability for Modelling CAPM Systems. *CAPM Project Team Working Paper*, Durham and Newcastle University.

Hofmann, A. G., Stanley, G. M. and Hawkinson, L. B., 1989, Object Oriented Models and Their Applications in Real-time Expert System. *Simulation and AI*, Proc. of the SCS Western Multiconference, San Diego, California, Simulation Series, Vol.20 No.3 July, 1989, pp.27-32.

Ignall, E.J., Kolesar, P. and Walker, W.E., 1978, Using Simulation To Develop and Validate Analytic Models: Some Case Studies. *Operations Research*, Vol.26, No.2, pp. 237-253.

Ip, W.H., 1993, The Application of Time Series Models in MRP Simulation. PhD Thesis, Loughborogh University of Technology.

Jenkins, G.M. and Watts, D.G., 1968, *Spectral analysis and its applications*. San Francisco, CA. Holden-Day.

Jiao Hong, 1991, Integrated Knowledge Based Hierarchical Modelling of Manufacturing Organizations. PhD Thesis, Loughborogh University of Technology.

Jorysz, H.R. and Vernadat, F.B., 1990a, CIM-OSA Part 1: Total Enterprise Modelling and Function View. *Int. J. Computer Integrated Manufacturing*, Vol.3, No.3 and 4, pp. 144-156.

Jorysz, H.R. and Vernadat, F.B., 1990b, CIM-OSA Part 2: Information View. *Int. J. Computer Integrated Manufacturing*, Vol.3, No.3 and 4, pp.157-167.

Jothishankar, M.C. and Wang, H.P., 1993, Metamodelling a Just-in-time Kanban System. *Int. J. Operation Production and Management*, Vol.13 No.8, pp.18-36.

Khoshnevis, B. and Chen, A.P., 1986, An Expert Simulation Model Builder. *Intelligent Simulation Environment (Eds. Luker, P.A. and Adelsberger, H.H.)*, Simulation Series, Vol.17 No.1, Proc. of Conf. on Intelligent Simulation Environment, 23-25 Jan. 1986, San Diego, California, USA. pp. 129-132.

Kleijnen, J.P.C., 1979, Regression Metamodels for Generalising Simulation Results. *IEEE Transactions on Systems, Man and Cybernetics*, Vol.9 No.2, pp.93-96.

Kleijnen, J.P.C., 1987, *Statistical Tools for Simulation Practitioners*, Marcel Dekker, New York, NY.

Klittich, M., 1990, CIM-OSA Part 3: CIM-OSA Integrating Infrastructure - The Operational Basis for Integrated Manufacturing Systems. *Int. J. Computer Integrated Manufacturing*, Vol.3, No.3 and 4, pp. 168-180.

Korson, T. and McGregor, J.D., 1990, Understanding OO: A Unifying Paradigm. *Communications of the ACM*, Vol.33, No.9, pp. 41-60.

Kreutzer, W., 1986, *System Simulation: Programming Style and Languages*, Addison-Wesley, pp.1-40.

Kuei, Chu-Hua and Madu, C.N., 1994, Polynomial Metamodelling and Taguchi Designs in Simulation With Application to the Maintenance Float System. *European Journal of Operational Research*, Vol.72, No.2, pp. 364-375.

Lal, H. and Peart, R. M., 1989, PROLOG For An Integrated Simulation and Expert System. *Advances in AI & Simulation*, Proc. of the SCS Multiconference on AI and Simulation, 28-31, March, 1989, Tampa, Florida, pp.6-11

Langen, P.A., 1987, Application of Artificial Intelligence Techniques to Simulation. *Simulation and AI (Eds. Luker, P.A. and Birtwistle, G.)*, Simulation Series, Vol.18 No.13, Proc. of Conf. on AI and Simulation, 14-16 Jan. 1987, San Diego, California, pp. 49-57.

Lavery, R.G., 1986, Artificial Intelligence and Simulation: An Introduction. *1986 Winter Simulation Conference Proceedings (Eds. Wilson, J.R., Henriksen, J.O. and Roberts, S.D.)*, Dec. 8-10, 1986, Washington, D.C., U.S.A., pp. 448-452.

Law, A. M., 1986, Introduction to Simulation: A Powerful Tool for Analyzing Complex Manufacturing Systems. *Industrial Engineering*, May, pp. 46-63.

Law, A. M. and W. D. Kelton, 1979, *Confidence Intervals for Steady-state Simulations*, I. A Surrey of Fixed Sample Size Procedures. Technical Report 78-5, Department of Industrial Engineering, University of Wisconsin, Op. sp 1.

Li, L. and Chiu, F., 1993, Menufactuirng Cell Operating Characteristics. *Europe Journal of Operational Research*, 69 pp. 424-437.

Lim, B.S., Marshall, G., Kellett, J.M., Kuczora, P.W. and Boardman, J.T., 1987, A foundation for a Knowledge Based Computer Integrated Manufacturing System. *Proc. 4ht European Conf. on Automated Manufacturing*, May, pp. 459-474.

Love, D.M., Barton, J.A. and Cope, N., 1992, Whole Business Simulation And Engineering Applications. *Eight International Conference Computer-Aided Production Engineering*, Edingburgh, August, pp. 166-173.

Mackulak, G.T., 1983, An Examination of the $IDEF_0$ Approach Used As A Potential Industry Standard for Production Control System Design. *Automated Manufacturing* (Ed. Gardner, L.B.), A Symposium Sponsored by ASTM Committee E-31 on Computerised Systems, 5-6 April, 1983, San Diego, California, pp. 136-149.

Madu, C.N. and Chanin, M.N., 1992, A Regression Metamodel of a Maintenance Float Problem with Erlang-2 Failure Distribution. *Int. J. Prod. Res.*, Vol.30, No.4, pp.871-875.

Maxwell, W.L., Muckstadt, J.A., Thomas, L.J. and Van der Eecken, J., 1983, A Modelling Framework For Planning and Control of Production in Discrete Parts Manufactuirng and Assembly Systems. *Interfaces*, Vol.13, pp.92-104.

Meisel, W.S. and Collins, D.C., 1973, Repro-modelling: an Approach To Efficient Model Utilisation and Interpretation. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-3 (4), pp. 349-358.

Meyer, W. and Isenberg, R., 1987, Aspects of Knowledge-based Factory Supervision System. *ESPRIT '87 Achievements and Impact*, Part 2, North-Holland, pp. 1671-1689.

Monarchi, D.E. and Puhr, G.I., 1989, A Research Typology for Object-oriented Analysis and Design. *Communications of the ACM*, Vol.35, No.9, pp.35-47.

Montgomery, D.C. and Peck, E.A., 1982, Introduction to Linear Regression Analysis, Wiley, New York.

Montgomery, D.C., 1991, *Design and Analysis of Experiments*, John Wiley & Sons.

Montgomery, D.C., 1992, Introduction to Linear Regression Analysis, Wiley, New York.

Mujtaba, M.S., 1994, Simulation Modelling of a Manufacturing Enterprise with Complex Material, Information and Control Flows. *Int. J. Computer Integrated Manufacturing*, Vol. 7, No. 1, pp. 29-46.

Neter, J. W. Wasserman and M. H. Kutner, 1985, *Applied Linear Statistic Models*. Second ed. Homewood, IL: Richard D. Irwin.

Newbold, P., 1981, Some Recent Developments in Time Series Analysis. *Int. Statist. Review*, Vol.49, pp. 53-66.

Newbold, P., 1984, Recent Developments in Time Series Analysis. *Int. Statist. Review*, Vol.52, pp. 183-192.

O'Keefe, 1986, Simulation and Expert Systems: A Taxonomy and Some Examples, *Simulation*, 46, pp.10-16.

Palaniswami, S. and Jenicke, L., 1992, A Knowledge-based Simulation System for Manufacturing Scheduling. *International Journal of Operational and Production Management*,Vol.12 No.11, pp. 4-14.

Parnaby, J., 1991, Design Effective Organizations. *Int. J. Technology Management*, Vol. 6, No. 1/2, pp.15-31.

Peck, A. C., 1989, Knowledge-based Approach to Modelling and Simulation. *Advances in AI & Simulation*, Proc. of the SCS Multiconference on AI and Simulation, 28-31, March, 1989, Tampa, Florida, pp.224

Priestley, M.B., 1981, *Spectral Analysis and Time Series*. New York, Academic Press.

Rajamani, D. and Singh, N., 1991, A Simulation Approach to the Design of An Assembly Line: A Case Study. *International Journal of Operational and Production Management*, Vol.11 No.6, pp. 66-75.

Rembold, U., Blume, C. and Dillmann, R., 1984, *Computer Integrated Manufactuirng Technology and Systems*. Springer-Verlag.

Rembold, U. and Nnaji, B., 1991, The Role Of Manufacturing Models For the Information Technology Of the Factory Of the 1990s. *Journal of Design and Manufacturing*, Vol.1, pp. 67-87.

Rotmas, J. and Vrieze, O.J., 1990, Metamodelling and Experimental Design: Case Study of the Green House Effect. *European Journal of Operational Research*, 47, pp. 317-329.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorensen, W., 1991, *Object-Oriented Modeling and Design*, chapter 2: Modeling As a Design Technique. Prentice-Hall.

Russell, E.C., 1986, SIMSCRIPT II.5 Tutorial. *1986 Winter Simulation Conference Proceedings (Eds. Wilson, J.R., Henriksen, J.O. and Robert, S.D.)*, Dec. 8-10, 1986, Washington, D.C., USA, pp 75-79.

Ryan, B.F., Joiner, B.L. and Ryan, T.Y., 1985, *MINITAB Handbook*. Second Edition, Boston: PWS-KENT Publishing Company.

Sarkis, and Li, L., 1994, A IDEF0 Functional Planning Model in Strategic Implementation of CIM System. *Int. J. Computer Integrated Manufacturing*, Vol.7, No.2, pp. 100-115.

SAS, 1985, *SAS® User's Guide:Statistics*. SAS Institute, Cary, NC.

Schriber, T.J., 1986, Introduction to GPSS. *1986 Winter Simulation Conference Proceedings (Eds. Wilson, J.R., Henriksen, J.O. and Robert, S.D.)*, Dec. 8-10, 1986, Washington, D.C., USA, pp 75-79.

Shannon, R.E., 1992, Introduction To Simulation. *Proceeding of the 1992 Winter Simulation Conference* ed. J.J. Swain, D.Goldsman, R.C.Crain and J.R. Wilson. pp.65-73.

Shanon, R.E., 1984, Keynote Address Artificial Intelligence and Simulation. *1984 Winter Simualtion Conference Proceedings* (Eds. Sheppard, S., Pooch, U.W. and Pegden, C.D.) Nov. 28-30, 1984, Dallas, Texas, pp.3-9.

Skooglund, N. S., 1989, Application of an AI Simulation Tool in the Design of a Manufacturing Process, *Advances in AI & Simulation*, Proc. of the SCS Multiconference on AI and Simulation, 28-31, March, Tampa, Florida, pp.109-113.

SofTech Inc., 1979, *IDEF Architect's Manual: ICAM Definition Method IDEF$_0$*.

SPSS-X User's Guide, 1986, McGraw-Hill.

Starbird, S.A., 1990, A Metamodel Specification for a Tomato Processing Plant. *Journal of the Operational Research Society (UK)*, Vol.41, No.3, pp. 229-240.

Starr, P.J., 1991, Integration of Simulation and Analytical Submodels for Supporting Manufacturing Decisions. *Int. J. Prod. Res.* Vol. 20, No.9, pp. 1733-1746.

Stefik, M and D. G. Bobrow, 1986, Object-Oriented Programming: Themes and Variations, *The AI Magazine*, Vol.6 No.4 Winter 1986, pp. 40-62.

Toni, A. D., Filippini, R. and Forza, C., 1992, Manufacturing Strategy in Global Markets: An Operations Management Model. *Int. J. Operations & Production Management*, Vol.12, No.4, pp. 7-18.

Tsai Wen-Hsien, 1992, Performance Measurement For Advanced Manufactuirng Technology (AMT). *The Second Conference on Automation Technology*, July, Taipei, Taiwan. Vol.4, pp. 13-20.

Venue, 1992, Venue Product Fact Sheets, *Venue, 1549 Industrial Road*, San Carlos, CA 94070, USA.

Wallace, R. and Mills, R.I., 1988, Simulation As An Essential Element Of Manufactuirng Strategy. *Proc. 4th Int. Conf. Simulation in Manufacturing*, November, pp. 227-237.

Wang, W., 1989, *A Knowledge Based Modelling System for the Design and Evaluation of Flexible Manufacturing Facilities*. Ph.D Thesis, Loughborough University of Technology.

Widman, L. E., K. A. Loparo and N. R. Nielsen, 1989, *Artifical Intelligence, Simulation, And Modelling*, John Wiley & Sons.

Wilson, J.M., 1991, Supplier Credit in the Economic Order Quantity Model. *International Journal of Operational and Production Management*, Vol.11 No.9, pp. 64-71.

Wortmann, J. C.(Ed.), 1994, CIM Architectures. *Computers In Industry*, Vol.24, No.2-3, pp. 109-266.

Wu, B., 1990, *Design of Manufacturing System*. publisher

Xerox, 1988, *Xerox LOOPS Reference Manual*. Lyric/Medley Release, Xerox PARC, July, USA.

Yu, B. and Popplewell, K., 1994, Metamodelling in Manufacturing: A Survey, *Int. J. Prod. Res.*, Vol.32, No.4, pp. 787-796.

Zulch, G. and Grobel, T., 1992, Simulating the Departmental Organization for Production To Order. *'ONE-OF-A-KIND' PRODUCTION: New Approaches*, (Eds. Hirsch, B.H. and Thoben, K.-D.) Elsevier Science Publishers B.V., Nother-Holland.

# Appendix I

# OBJECT-ORIENTED PROGRAMMING TECHNIQUES

This appendix introduces the object-oriented programming paradigm applied in LISP. The aim is to offer some basic knowledge of object-oriented programming to those readers, who are willing to know more about the LOOPS environment and are interested to use this technique.

## I.1 Object-Oriented Programming

Object oriented programming has been used to mean different things, but there is one thing in common, that is object. An object is defined as a symbol associated with a unique database of properties and operations which represent the object (Stefik and Bobrow, 1986). In programming, an object is defined by its attributes which are called instance variables. If several objects share the same attributes, these attributes can be defined as a class variable. Objects communicate with each other by the passing of messages, which carry the specification of an operation an object is required to perform and will responded to. To react to the received message the object uses its own procedures called methods for performing operations. Hence, message sending is the key that triggers all of the actions in object-oriented programming.

Message sending also supports an important principle in programming: data abstraction (Stefik and Bobrow, 1986). This principle is that calling programs should

not make assumptions about the implementation and internal representations of data types. This means that message specifies only which operation should be performed, not how the operation should be performed. From the principle, one of advantages of object oriented programming can be seen is that changing the underlying implementation is possible without changing the calling programs. This expresses strong localisation of information in closed modules is strictly adhered to (Lal and Peart, 1989) (Skooglund, 1989). Other data structures, functions, procedures are private and can only be used from within an object (Kreutzer, 1986).

Another feature of object-oriented programming is specialisation. Because objects are organised into hierarchical classes in which subclasses inherit properties and message patterns from their super class, it enables the easy creation of objects that are almost like other objects with specialisation, or say, with adding in a few changes. Inheritance permits the representation of knowledge in a way which minimises redundancy and simplifies data updating and modification easily.

It is the abstraction capability that complicated models in object-oriented paradigms can be built rapidly and reliably (Hofmann et al., 1989). This provides enough flexibility so that the user can specify the behaviour of individual instances (Peck, 1989) (Begg and Worsley, 1989). Therefore, it is attractive for most researchers to use it as a vehicle for constructing simulation models of real world systems.

## I.2 Object Oriented Programming in LOOPS

LOOPS, stands for Lisp Object Oriented Programming System, integrates four programming paradigms: procedure-oriented programming, object-oriented programming, access-oriented programming and rule-oriented programming. The integration has two major themes in LOOPS: first is to allow the different paradigms to be used together in building a knowledge system; and second is the integration of a programming environment for creating and debugging a knowledge system. In the first integration, either the rule or the ruleset is the object. Methods kept in classes can be either lisp function or rulesets. The procedures in active values can be lisp functions, rulesets or can call on methods. In the second integration in LOOPS, this has led to the same synergy that is exploited in using multiple paradigms for application programs. The following is an example which shows the specification of the object Monitor in the LOOPS environment .

```
[Monitor
 ((Meta Classes AbstractClass
        Edited (*; " Edited 29-Feb-89 by P.Chang)
        doc ( ** This sample class illustrates the syntax of classes in Loops)
  (Supers Simulation-Object)
  (Classes Variables)
        (specificItems ((Attach AttachToObject doc "Attach self to an object")
                        (Detach Detach doc "Detach self from the object")
                        (Reattach Reattach doc "Reattach self to the object"))
        (image SetBitmap doc "a bit map used to represent self on a map"))
  (Instance Variables
```

```
(reattach? NIL doc "T if self should be reattached when the display
        window is opened")
(attachee NIL doc "an list of objects we are attached to")
(timeOn NIL doc "the time relative to now when self will switch on")
(timeOff NIL doc "the time relative to the timeOn if self is off or
        relative to now if self is on when self will switch"))
(Methods
    MONITOR.AttachToObject  MONITOR.Detach  MONITOR.Reattach
    MONITOR.SwitchOn MONITOR.SwitchOff MONITOR.ChangeName
    MONITOR.Reset MONITOR.Remove)]
```

Every member of the class may have a different value for an instance variable, such as the location instance variable. However each instance of the class has the same class variable, like *image* in the class variable.

In any traditional programming language, procedural code operating over data structures is applied. For example, in modelling the behaviour of monitor, a set of procedures would be defined which would operate over some data structure (a state vector) which represents a particular monitor.

In an object oriented system, the set of procedures applicable to a particular state vector are associated with that state vector. The conventional call and return function protocol is replaced by a slightly different "message send and return" protocol. Here a message is sent to an object consisting of a selector identifying what the user wants to

take place and message arguments. Thus, in LOOPS the behaviour of a monitor would be defined separately for each type of monitor and it would be involved in sending a message to the relevant objects from where the value of the variables to be monitored can be obtained. In LOOPS code, this would be:

(- monitor Switch).

Thus an immediate advantage of object oriented programming is the reduction in the number of decision points present in the code. In an object oriented system, code is located in a manner that clearly associates it with the data structure over which the code is intended to operate (Lal and Peart, 1989). The functional system does not need to be changed when new types of monitor are introduced or when the data representations are altered. This makes the system much easier to modify and thus supports a prototyping approach to software development.

The behaviour of objects are specified by the functions that are invoked as a result of the message sent. Objects of different types may use different functions in response to the same message. For example

(- monitor Switch).

could call on of following:

SingleAttribute.Switch

MultiAttribute.Switch

depending on whether the monitor is a single attribute or a multi attribute monitor.

Objects of the same type will always use the same function in response to the same selector but they may have different data over which the function operates. For example, two single attribute monitors could have different attributes to attach so (-monitor Switch) could result in one string output and another in numerical data output.

Although the data values on different objects of similar type may vary, the actual data structure needs to be identical otherwise the functions applicable to that type of object might attempt to access non-existent data.

# Appendix II

## The Metamodelling Program

The hybrid modelling program consist of a suite 7 pieces of source code, of which two examples are GENERATOR.SRC and COLLECTOR.SRC enclosed in this Appendix. GENERATOR.SRC is used for construction of a metamodel, and COLLECTOR.SRC is used for collecting user specified simulation results.

No matter what function a collector or a zoom node performs, each of them has been treated as an object in the LOOPS and LISP environment. Therefore, each of them is a Class. Within the GENERATOR.SRC source code, under **CLASSES** is MetamodelGenerator. Each object respond to the message its received by using a methods, which is equivalent to the procedure in traditional programming. All the methods are possessed by an object is listed under the **METHODS** in the first part of the description of the object. In GENERATOR, it has ChooseData, Formulation, Validating and so on.

Each object also have variables to store data. The variables shared between different instances are defined as **ClassVariables**, such as specificItems on the GENERATOR object. The values used locally and used to explicit instances are defined as **InstanceVariables**, for example image-bm (icon bitmap).

To collect simulation data, an independent variable menu appears on the screen. This is caused by calling COLLECTOR.ChooseVarMenu. Similar to COLLECTOR, once a user select "Formulate metamodel" from the GENERATOR object menu, GENERATOR class responds to this by operating GENERATOR.Formulation. A method can be a primitive method or can invoke another method within. For instance, in METHOD GENERATOR.ChooseData, where a part of code likes:

```
(MENU (_ self GetChooseDataMenu))
```

(_ self GetChooseDataMenu) shows another method, named GetChooseDataMenuand and stores this information on the object itself (self), is called. When the primitive method is reached, the actual activities starts.

This structure is utilised for each of the pieces of code, and the following examples illustrate the object classes for the work.

```
(DEFINE-FILE-INFO  READTABLE "XCL"  PACKAGE "INTERLISP")
  (FILECREATED "16-Nov-94 15:39:57"   |{DSK}<home>jelly>hmsm>bing>tem>metacode>GENERATOR.SRC;11|
29109

      |changes| |to:|   (METHODS |MetamodelGenerator.GetTypeMethodInfo|
                                |MetamodelGenerator.Validating|)
                         (FNS |MetamodelGenerator.GetTypeMethodInfo|)
                         (CLASSES |MetamodelGenerator|)
                         (VARS GENERATORCOMS)

      |previous| |date:| " 2-Nov-94 16:43:42"
|{DSK}<home>jelly>hmsm>bing>tem>metacode>latestcode>GENERATOR.SRC;2|)


(PRETTYCOMPRINT GENERATORCOMS)

(RPAQQ   GENERATORCOMS
       ((CLASSES |MetamodelGenerator|)
        (METHODS |MetamodelGenerator.ChooseData| |MetamodelGenerator.ClearLogInformation|
                 |MetamodelGenerator.CloseLog| |MetamodelGenerator.ConditionalFormu|
                 |MetamodelGenerator.CreateMatrix| |MetamodelGenerator.DSPANOVA|
                 |MetamodelGenerator.DeleteLog| |MetamodelGenerator.DescribeCurrentLog|
                 |MetamodelGenerator.EntryData| |MetamodelGenerator.Formulation|
                 |MetamodelGenerator.Validating| |MetamodelGenerator.Formul|
                 |MetamodelGenerator.GetChooseDataMenu| |MetamodelGenerator.GetFromLog|
                 |MetamodelGenerator.GetTypeMethodInfo| |MetamodelGenerator.GetTypeOfData|
                 |MetamodelGenerator.ObtainData| |MetamodelGenerator.OpenLog|
                 |MetamodelGenerator.ReadBriefDescription| |MetamodelGenerator.ReadDataInMatrix|
                 |MetamodelGenerator.RestoreFileData| |MetamodelGenerator.SelectLog|)))

(DEFCLASSES |MetamodelGenerator|)
(DEFCLASS   |MetamodelGenerator|
   (|MetaClass| |Class| |Edited:|                                  (* \; "Edited 14-Nov-94 10:31 by bing")
         )
   (|Supers| |NodeGraphic| SIMULATION-OBJECT)
   (|ClassVariables| (|specificItems| #,($A (("Describe current data file" |DescribeCurrentLog|
"Prints the brief description associated with the current log" NIL NIL |HasLog|) ("Select file"
|SelectLog| "Lets you choose from the log indices which log you want to analyse" NIL |HasLog|
(NOT |HasLog|)) ("Change file" |SelectLog| "Lets you change the log you want to analyse" NIL NIL
|HasLog|) ("Formulate metamodel" |Formul| "Prints a summary of the logged information" NIL NIL
|HasLog|) ("Validate the metamodel" |Validating| "Validate the metamodel which is latest
generated" NIL NIL |HasLog|) ("Delete file" |DeleteLog| "Deletes the current log permanently from
disc" NIL NIL |HasLog|)) |UnionSuperValue| NIL)
                              |doc| "the specific items for this class")
         (|typeSummaryMethods| (((SMALLP FIXP FLOATP BIGNUM)
                                 |\\PrintNumericVarSummary|
                                 (("Construct metamodel (whole)" '|Formulation|
                                        "help string for construct metamodel"
                                        (SUBITEMS ("Construct metamodel (part)"
                                                   '|ConditionalFormu|)))
                                  ("Transfermation" '|Transfl| "help string for construct
metamodel"
                                        (SUBITEMS ("Expotential" '|PotientialTranf|)
                                                  ("Square" '|SquareTranf|)
                                                  ("1/Y" '|Transfl|)))
                                  ("Display ANOVA" 'DSPANOVA
                                        "Plots number of times each value taken on ignoriing
time")
                                  ))
                                ((LITATOM STRINGP CHARACTER)
                                 |\\PrintHistSummary| |\\PlotHistIVData|))
                  |doc| "lst used to identify the method which will be used to print the summary
of the particular type of stored data - all methods are passed the stream for printing and the
varData of the form (<varName> . stream) - also holds plotting methods appropriate for varData"
                  ))
   (|InstanceVariables| (|briefDescription| NIL |DontSave| |Any| |doc|
                                 "current log's brief description string")
         (|indexFile| NIL |doc| "the file which is used as the index for the current log")
         (|log| NIL |DontSave| |Any| |doc|
                  "log index of the file streams containing the logged information")
         (|matrix| NIL |doc|
                  "the matrix that has been used or is going to be used for metamodel
construction.")
         (|window| NIL |doc| "edit and display window")
         (|predThreshold| 90 |sample#| 10 |doc| "threshold is percentage of elements in data
that will lead to data being treated as matching the predicate type sample# is the sample size
used to establish the percentage"
                  )
         (|image-bm| ANALYSER-BM |doc| "bit map used to represent self on a map")))
```

```
(|\\BatchMethodDefs|)
(METH |MetamodelGenerator|    |ChooseData| NIL
      "Let the user choose the data that he wants to do something with and reture that data."
      (|category| (|MetamodelGenerator|)))

(METH |MetamodelGenerator|    |ClearLogInformation| NIL
      "clear the log information from self" (|category| (|MetamodelGenerator|)))

(METH |MetamodelGenerator|    |CloseLog| NIL
      "Close the log file if they are open." (|category| (|MetamodelGenerator|)))

(METH |MetamodelGenerator|    |ConditionalFormu| NIL
      "construct metamodel with user selected independent variables" (|category| (

|MetamodelGenerator|
                                                                                  )))

(METH |MetamodelGenerator|    |CreateMatrix| (|data|)
      "Create a matrix, and then put data into it." (|category| (|MetamodelGenerator|)))

(METH |MetamodelGenerator|    DSPANOVA NIL
      "display the results of Analysis of Variance" (|category| (|MetamodelGenerator|)))

(METH |MetamodelGenerator|    |DeleteLog| NIL
      "Delete the current log completely!" (|category| (|MetamodelGenerator|)))

(METH |MetamodelGenerator|    |DescribeCurrentLog| NIL
      "prints out the brief description of the current data file if any" (|category| (

|MetamodelGenerator|
                                                                                  )))

(METH |MetamodelGenerator|    |EntryData| (|data|)
      "Prompt metamodel editing menu here or not? (considering)" (|category|
(|MetamodelGenerator|)))

(METH |MetamodelGenerator|    |Formulation| NIL
      "To construct metamodel from the file." (|category| (|MetamodelGenerator|)))

(METH |MetamodelGenerator|    |Validating| NIL
      "Validate metamodel with Double-Cross methods" (|category| (|MetamodelGenerator|)))

(METH |MetamodelGenerator|    |Formul| (|IVData|)
      "metamodel construction and display selection." (|category| (|MetamodelGenerator|)))

(METH |MetamodelGenerator|    |GetChooseDataMenu| NIL
      "Return a menu which will return a varData appropriate part of the log struture under user
selection"
      (|category| (|MetamodelGenerator|)))

(METH |MetamodelGenerator|    |GetFromLog| (|keys|)
      "Return the fileStream containing the data stored under keys in the log (if any)"
(|category|
                                                                                  (

|MetamodelGenerator|
                                                                                  )))

(METH |MetamodelGenerator|    |GetTypeMethodInfo| (|varData|)
      "Method documentation" (|category| (|MetamodelGenerator|)))

(METH |MetamodelGenerator|    |GetTypeOfData| (|varData|)
      "Method documentation" (|category| (|MetamodelGenerator|)))
```

```
(METH |MetamodelGenerator|     |ObtainData| (|varData|)
      "Read out data from saved file." (|category| (|MetamodelGenerator|)))

(METH |MetamodelGenerator|     |OpenLog| NIL
      "Open the log file" (|category| (|MetamodelGenerator|)))

(METH |MetamodelGenerator|     |ReadBriefDescription| (|logIndexFile|)
      "Read and return the brief description found on the logIndexFile" (|category| (

|MetamodelGenerator|
                                                                                  )))

(METH |MetamodelGenerator|     |ReadDataInMatrix| (|data| |tags?|)
      "To allocate data to each element of the matrix." (|category| (|MetamodelGenerator|)))

(METH |MetamodelGenerator|     |RestoreFileData| (|file|)
      "Make the log index on self consistent with the stored information on file" (|category| (

|MetamodelGenerator|

)))

(METH |MetamodelGenerator|     |SelectLog| NIL
      "Let user to choose a new logger to work on" (|category| (|MetamodelGenerator|)))


(|Method| (  (|MetamodelGenerator| |ChooseData|)   |self|)
   "Let the user choose the data that he wants to do something with and reture that data."
   (  |if|    (@ |:log|)
         |then|    (MENU (_ |self| |GetChooseDataMenu|))
       |else|    (_ |self| |PrintMessage| "No items present in collecter.")
          NIL))

(|Method| (  (|MetamodelGenerator| |ClearLogInformation|)   |self|)
   "clear the log information from self"
   (  |self| |CloseLog|)
   (_@ |:briefDescription| NIL)
   (_@ |:indexFile| NIL)
   (_@ |:log| NIL)
   (_ |self| |ChangeState| NIL NIL)
   NIL)

(|Method| (  (|MetamodelGenerator| |CloseLog|)   |self|) "Close the log file if they are open."
   (  |if|    (@ |:log|)
         |then|    (|OperateOnLogFiles| (@ |:log|)
                        (FUNCTION (LAMBDA (|file|)
                                     (  |if|    (AND (STREAMP |file|)
                                                      (OPENP |file|))
                                        |then|    (CLOSEF |file|)
                                      |else|    |file|)))
                        T))
   (_@ |:log:,chooseMenu| NIL)
   T)

(|Method| (  (|MetamodelGenerator| |ConditionalFormu|)   |self|)
   "construct metamodel with user selected independent variables"
   (LET ((|fileAndStream| (CAR (@ |:log|)))
         (|independentVars| (CAAR (@ |:log|))))
        (  |if|    |fileAndStream|
           |then|                                                (* \; "process the data")
               (LET ((|data| (_ |self| |ObtainData| |fileAndStream|)))
                    (  |if|    |data|
                          |then|    (_ |self| |CreateMatrix| |data|)
                       |elseif|    (@ |:matrix|)
                          |then|    (_ |self| |PrintMessage| "Using same data file")))
               (  |if|    (AND (@ |:matrix|)
                                (NULL (@ |:window|)))
                  |then|

                        (* |;;|
                  "data has been read in. In sequence, to create metamodel edit window
instance.")
```

```
                    (LET ((|MetamodelEDWindow| (_ ($ |MetamodelWindow|)
                                                  |New|)))
                         (SETQ \v (_ (@ |:drawing|)
                                     |PromptForWord|
                                     "How many independent variables are you interested?
"))
                         (_@ |MetamodelEDWindow| |colNumList| NIL)
                         (  |for|   \i   |from|   1   |to|   \v
                              |do|   (_@ |MetamodelEDWindow| |colNumList|
                                          (ATTACH (_ (@ |:drawing|)
                                                     |PromptForWord|
                                                     (CONCAT
                                                           "Input column number of
the "
                                                            \i " independent
variable: "
                                                            ))
                                                  (@ |MetamodelEDWindow|
|colNumList|))))
                                 (_@ |:window| |MetamodelEDWindow|)
                                 (_@ |MetamodelEDWindow| |toGenerator| |self|)
                                 (_@ |MetamodelEDWindow| |dataTab| (@ |:matrix|))
                                 (_ |MetamodelEDWindow| |SetVariableName| |independentVars|)
                                 (_ |MetamodelEDWindow| |Start| T))
                        |elseif|   (@ |:matrix|)
                          |then|

                                (* |;;|
                                "metamodel edit window is existing, so pass the new data to the
instance")

                    (LET ((|MetamodelEDWindow| (@ |:window|)))
                         (SETQ \v (_ (@ |:drawing|)
                                     |PromptForWord|
                                     "How many independent variables are you interested?
"))
                         (_@ |MetamodelEDWindow| |colNumList| NIL)
                         (  |for|   \i   |from|   1   |to|   \v
                              |do|   (_@ |MetamodelEDWindow| |colNumList|
                                          (ATTACH (_ (@ |:drawing|)
                                                     |PromptForWord|
                                                     (CONCAT
                                                           "Input column number of
the "
                                                            \i " independent
variable: "
                                                            ))
                                                  (@ |MetamodelEDWindow|
|colNumList|))))
                                 (_ |MetamodelEDWindow| |OpenWindow|)
                                 (_@ |MetamodelEDWindow| |dataTab| (@ |:matrix|))
                                 (_ |MetamodelEDWindow| |SetVariableName| |independentVars|)
                                 (_ |MetamodelEDWindow| |Start| T)))
                    |else|                                   (* \; "do nothing")
                        (_ |self| |PrintMessage| "Unable to formulate metamodel"))
                NIL))

(|Method| (  (|MetamodelGenerator| |CreateMatrix|)   |self| |data|)
   "Create a matrix, and then put data into it."
   (  |if|   |data|
        |then|   (LET ((|independentVars| (CAAR (@ |:log|)))
                        (|rows| (LENGTH |data|))
                 |colums| |matrix| |tags?|)
                (  |if|   (LISTP |independentVars|)
                     |then|

                            (* |;;|
                            "create a matrix where independent variable is more than one")

                         (  |if|   (OR (EQ (LENGTH (CAR |data|))
                                           5)
                                       (EQ (LENGTH (CAR |data|))
                                           6))
                              |then|                    (* \;
                                                      "there are two tags on the transition
map")
                                   (SETQ |colums| (IPLUS (LENGTH |independentVars|)
                                                         1))
                                   (SETQ |tags?| NIL)
                            |elseif|   (OR (EQ (LENGTH (CAR |data|))
```

```
                                                    7)
                                                (EQ (LENGTH (CAR |data|))
                                                    8))
                                |then|                    (* \;
                                                "there are three tags on the transition
map")
                                    (SETQ |colums| (IDIFFERENCE (LENGTH (CAR |data|))
                                                                1))
                                    (SETQ |tags?| T))
                            |else|
                                (* |;;| "independent varible is WIP only")

                                (  |if|   (EQ (LENGTH (CAR |data|))
                                              4)
                                    |then|   (SETQ |colums| 2)
                                             (SETQ |tags?| NIL)
                                  |elseif|   (EQ (LENGTH (CAR |data|))
                                                 6)
                                    |then|   (SETQ |colums| 5)
                                             (SETQ |tags?| T)))
                            (_@ |:matrix| (|table| |rows| |colums| 'FLOATP 0))
                            (_ |self| |ReadDataInMatrix| |data| |tags?|))
        |elseif|   (@ |:matrix|)
          |then|   (_ |self| |PrintMessage| "Using previous data.")
        |else|   (ERROR "There has no data to build metamodel.")))

(|Method| (  (|MetamodelGenerator| DSPANOVA)   |self|) "display the results of Analysis of Variance"
   (LET ((|instance| (@ |:window|)))
        (_ |instance| DSPANOVA)))

(|Method| (  (|MetamodelGenerator| |DeleteLog|)   |self|) "Delete the current log completely!"
   (  |if|   (MOUSECONFIRM "About to delete all the current data file permanently!" NIL
                  (|_Obj|
                   |:drawing| |GetPromptWindow|))
         |then|   (RESETLST
                      (RESETSAVE (CURSOR WAITINGCURSOR))
                      (_ |self| |CloseLog|)
                      (|OperateOnLogFiles| (@ |:log|)
                            'DELFILE
                            (DELFILE (@ |:indexFile|))
                            (_ |self| |ClearLogInformation|)
                            (_ |self| |ChangeState| NIL NIL)))))

(|Method| (  (|MetamodelGenerator| |DescribeCurrentLog|)   |self|)
   "prints out the brief description of the current data file if any"
   (  |if|   (@ |:briefDescription|)
         |then|   (_ |self| |PrintMessage| (@ |:briefDescription|))
        |else|   (_ |self| |PrintMessage| "No brief description available")))

(|Method| (  (|MetamodelGenerator| |EntryData|)   |self| |data|)
   "Prompt metamodel editing menu here or not? (considering)"
   (LET* ((|selectors| (CADDR (_ |self| |GetEditMenu| |data|)))
          (|selector| (  |if|   (LISTP |selectors|)
                            |then|   (MENU (  |create|   MENU
                                                ITEMS _ |selectors|
                                                TITLE _ "Metamodel Edit Menu"))
                            |else|   |selectors|)))
         (  |if|   |selector|
               |then|   (CL:BREAK)
             |else|                                          (* \;
                                                   "Let user know we can't do anything")
                  (_ |self| |PrintMessage| "Unable to plot information")
                  NIL)))

(|Method| (  (|MetamodelGenerator| |Formulation|)   |self|) "To construct metamodel from the file."
   (LET ((|fileAndStream| (CAR (@ |:log|)))
         (|independentVars| (CAAR (@ |:log|))))
         (  |if|   |fileAndStream|
             |then|                                          (* \; "process the data")
                 (LET ((|data| (_ |self| |ObtainData| |fileAndStream|)))
                      (_ |self| |CreateMatrix| |data|))
                 (  |if|   (AND (@ |:matrix|)
                                (NULL (@ |:window|)))
                      |then|

                            (* |;;|
                      "data has been read in. In sequence, to create metamodel edit window
instance.")
```

```
                (LET ((|MetamodelEDWindow| (_ ($ |MetamodelWindow|)
                                              |New|)))
                     (_@ |:window| |MetamodelEDWindow|)
                     (_@ |MetamodelEDWindow| |toGenerator| |self|)
                     (_@ |MetamodelEDWindow| |dataTab| (@ |:matrix|))
                     (_ |MetamodelEDWindow| |SetVariableName| |independentVars|)
                     (_ |MetamodelEDWindow| |Start| NIL))
        |elseif|     (@ |:matrix|)
          |then|
                      (* |;;|
                      "metamodel edit window is existing, so pass the new data to the
instance")
                    (LET ((|MetamodelEDWindow| (@ |:window|)))
                         (_ |MetamodelEDWindow| |OpenWindow|)
                         (_@ |MetamodelEDWindow| |dataTab| (@ |:matrix|))
                         (_ |MetamodelEDWindow| |SetVariableName| |independentVars|)
                         (_ |MetamodelEDWindow| |Start| NIL)))
          |else|                                          (* \; "do nothing")
              (_ |self| |PrintMessage| "Unable to formulate metamodel"))
      NIL))

(|Method| (  (|MetamodelGenerator| |Validating|)   |self|)
  "Validate metamodel with Double-Cross methods"
  (  |if|   (@ |:window|)
       |then|   (_ (@ |:window|)
                   |Validating|)
     |else|   (_ (@ |:drawing|)
                 |PrintMessage| "Construct metamodel first."))

(|Method| (  (|MetamodelGenerator| |Formul|)   |self| |IVData|)
  "metamodel construction and display selection."
  (LET* ((|selectors| (CADDR (_ |self| |GetTypeMethodInfo| |IVData|)))
         (|selector| (  |if|   (LISTP |selectors|)
                        |then|   (MENU (  |create|   MENU
                                          ITEMS _ |selectors|
                                          TITLE _ "Select options"))
                        |else|   |selectors|)))
        (  |if|   |selector|
             |then|   (_! |self| |selector| |IVData|)
           |else|   (_ |self| |PrintMessage| "Unable to plot information" NIL))))

(|Method| (  (|MetamodelGenerator| |GetChooseDataMenu|)   |self|)
  "Return a menu which will return a varData appropriate part of the log struture under user
selection"
  (  |if|   (  |type?|   MENU (@ |:log:;chooseMenu|))
       |then|   (@ |:log:;chooseMenu|)
     |else|   (_@ |:log:;chooseMenu|)
         (|\\MakeChooseDataMenu| (@ |:log|))))

(|Method| (  (|MetamodelGenerator| |GetFromLog|)   |self| |keys|)
  "Return the fileStream containing the data stored under keys in the log (if any)"
  (|GetMultiAssoc| |keys| (@ |:log|)))

(|Method| (  (|MetamodelGenerator| |GetTypeMethodInfo|)   |self| |varData|) "Method documentation"
  (  |bind|   (|types| _ (_ |self| |GetTypeOfData| |varData|))   |find|   |entry|
     |in|   (@ |::typeSummaryMethods|)   |suchthat|   (  |for|   |type|   |inside|   |types|
                                                          |always|   (FMEMB |type| (CAR |entry|))
                                                          )))

(|Method| (  (|MetamodelGenerator| |GetTypeOfData|)   |self| |varData|) "Method documentation"
  (LET ((|sample#| (@ |:predThreshold:,sample#|))
        (|done#| 0)
        |types|)
       (  |if|   (|ListLonger?| |types| 1)
            |then|   |types|
          |else|   (CAR |types|))))

(|Method| (  (|MetamodelGenerator| |ObtainData|)   |self| |varData|) "Read out data from saved file."
  (PROG (|dataToDeal|)
        (SETQ |dataToDeal| (MAP-META-DATA (CDR |varData|)))
        (RETURN |dataToDeal|)))

(|Method| (  (|MetamodelGenerator| |OpenLog|)   |self|) "Open the log file"
  (  |if|   (@ |:log|)
       |then|   (|OperateOnLogFiles| (@ |:log|)
                    (FUNCTION (LAMBDA (|file|)
                                  (  |if|   (STREAMP |file|)
                                        |then|   |file|
```

```
                                    |else|   (OPENSTREAM |file| 'INPUT 'OLD))))
                        T)))

(|Method| (  (|MetamodelGenerator| |ReadBriefDescription|)   |self| |logIndexFile|)
   "Read and return the brief description found on the logIndexFile"
   (LET ((|fileStream| (OR (OPENP |logIndexFile|)
                           (OPENSTREAM |logIndexFile'INPUT| 'OLD))))
        (PROG1 (READ |fileStream| STEM-RDTBL)
               (CLOSEF |fileStream|))))

(|Method| (  (|MetamodelGenerator| |ReadDataInMatrix|)   |self| |data| |tags?|)
   "To allocate data to each element of the matrix."
   (LET* ((|matrix| (@ |:matrix|))
          (|rows| (ARRAYSIZE |matrix|))
          (|colums| (ARRAYSIZE (ELT |matrix| 1))))
        (  |if|   (NULL |tags?|)
             |then|   (  |for|   \i   |from|   1   |to|   |rows|
                          |do|   (LET* ((|valueGroup| (CAR (NTH |data| \i)))
                                        (|depValue| (CAR (LAST |valueGroup|))))
                                       (SETM |matrix| \i 0 |depValue|)
                                       (  |for|   \j   |from|   1   |to|   (IDIFFERENCE |colums| 1
                                                                                )
                                          |do|   (SETQ |indValue| (CAR (NTH |valueGroup|
                                                                              (IPLUS \j 1))))
                                                 (SETM |matrix| \i \j |indValue|))))
             |else|   (  |for|   \i   |from|   1   |to|   |rows|
                          |do|   (LET* ((|valueGroup| (CAR (NTH |data| \i)))
                                        (|depValue| (CAR (LAST |valueGroup|))))
                                       (SETM |matrix| \i 0 |depValue|)
                                       (  |for|   \j   |from|   1   |to|   (IDIFFERENCE |colums| 1)
                                          |do|   (SETQ |indValue| (CAR (NTH |valueGroup|
                                                                              (IPLUS \j 1))))
                                                 (SETM |matrix| \i \j |indValue|)))))
        (_@ |:matrix| |matrix|)))

(|Method| (  (|MetamodelGenerator| |RestoreFileData|)   |self| |file|)
   "Make the log index on self consistent with the stored information on file"
   (  |self| |CloseLog|)
   (LET ((|stream| (OPENSTREAM |file| 'INPUT 'OLD)))

  (* |;;;| "Remember the index file")

        (_@ |:indexFile| |file|)

  (* |;;;| "read in the description")

        (_@ |:briefDescription| (READ |stream| STEM-RDTBL))

  (* |;;;| "Read in the log")

        (_@ |:log| (READ |stream| STEM-RDTBL))

  (* |;;;| "close the stream")

        (CLOSEF |stream|)

  (* |;;;| "Open the log")

        (_ |self| |OpenLog|)

  (* |;;;| "Inform user")

        (_ |self| |PrintMessage| (CONCAT "Metamodel generator now contains data from collecter
file "
                                         |file|)
                T T)
        T))

(|Method| (  (|MetamodelGenerator| |SelectLog|)   |self|)
   "Let user to choose a new logger to work on"
   (LET* ((|logIndexFiles| (DIRECTORY (PACKFILENAME 'NAME (@ |:controller:thisID|)
                                                     'EXTENSION
                                                     'IDEM-META-INDEX)))
          (|menuItems| (AND |logIndexFiles| (NCONC1 (  |for|   |file|   |in|   |logIndexFiles|
                                                       |collect|   (LIST (PACKFILENAME 'HOST NIL
                                                                                'DIRECTORY NIL
                                                                                'BODY |file|)
                                                                       (LIST
'|RestoreFileData|
                                                                                |file|)
```

```
                                                                                NIL))
                                                                  '("* None *" |ClearLogInformation|
                                                                        "Clears the collecter information from
self"
                                                                         )))))
        (  |if|    |menuItems|
           |then|    (STATE-MENU (_ |self| |CreateStateMenu| |menuItems| "Select saved data file")
                                )
            |else|    (_ |self| |PrintMessage| "No metamodel data files found on the disc")
               NIL)))

(|\\UnbatchMethodDefs|)
(DECLARE\: DONTCOPY
  (FILEMAP (NIL)))
STOP
```

```
(DEFINE-FILE-INFO  READTABLE "XCL"  PACKAGE "INTERLISP")
  (FILECREATED "17-Feb-95 11:24:43"   |{DSK}<home>jelly>hmsm>bing>tem>metacode>COLLECTER.SRC;32|
99243

      |changes| |to:|   (METHODS |Collecter.SwitchOn|)
                         (CLASSES |Collecter|)

      |previous| |date:| "17-Feb-95 10:48:27"
|{DSK}<home>jelly>hmsm>bing>tem>metacode>COLLECTER.SRC;31|)

; Copyright (c) 1994, 1995 by B.YU.  All rights reserved.

(PRETTYCOMPRINT COLLECTERCOMS)

(RPAQQ   COLLECTERCOMS
       ((CLASSES |Collecter|)
        (METHODS |Collecter.AttachIdemNode| |Collecter.AttachPartialZoomNode|
                 |Collecter.AttachZoomNode| |Collecter.ChooseAnObj| |Collecter.ChooseAnVariable|
                 |Collecter.ChooseMultiPairPath| |Collecter.ChoosePairOfPath|
|Collecter.ChooseVarMenu|
                 |Collecter.ChooseVariables| |Collecter.ConvertTypeToNumber| |Collecter.DSPWIPHist|
                 |Collecter.DettachAllVariables| |Collecter.DettachVariable|
|Collecter.EventChanged|
                 |Collecter.ExpungeData| |Collecter.GetMetaDescription|
                 |Collecter.GetMetaIndexFileName| |Collecter.GetQLength| |Collecter.GetQLength2|
                 |Collecter.GetWIPData| |Collecter.InitialLog| |Collecter.PickPath|
                 |Collecter.PickPaths| |Collecter.ProcessData| |Collecter.ProcessEvent|
                 |Collecter.PutToLog| |Collecter.ReSet| |Collecter.Record| |Collecter.RestData|
                 |Collecter.SPIndication| |Collecter.SSIndication| |Collecter.SState?|
                 |Collecter.SaveData| |Collecter.SaveDataIfWanted| |Collecter.Start|
                 |Collecter.StoreValue| |Collecter.SwitchOff| |Collecter.SwitchOn|
                 |Collecter.TakeRecord| |Collecter.\\GetFileStream| |Collecter.\\GetLogFileName|)
        (FNS |Collecter.GetQLengt|)))

(DEFCLASSES |Collecter|)
(DEFCLASS   |Collecter|
   (|MetaClass| |Class| |Edited:|                                  (* \; "Edited 17-Feb-95 10:39 by enby")
         )
   (|Supers| MULTIPLE-ATTRIBUTE-MONITOR)
   (|ClassVariables| (|specificItems| #,($A (("Switch On" |SwitchOn| "Will switch this monitor
on" (SUBITEMS ("Set time to switch on" |SetOnTime| "Allows you to enter a time when this monitor
will automatically switch on" NIL (|TimeOnSet?| T) (|TimeOnSet?| NIL)) ("Change time to switch
on" |SetOnTime| "Allows you to alter the time when this monitor will automatically switch on" NIL
NIL (|TimeOnSet?| T)) ("Set time to switch off" |SetOffTime| "Allows you to enter a time when
this monitor will automatically switch off" NIL (|TimeOffSet?| T) (|TimeOffSet?| NIL)) ("Change
time to switch off" |SetOffTime| "Allows you to alter the time when this monitor will
automatically switch off" NIL NIL (|TimeOffSet?| T))) O\n |Off|) ("Switch Off" |SwitchOff| "Will
switch this monitor on" (SUBITEMS ("Set time to switch off" |SetOffTime| "Allows you to enter a
time when this monitor will automatically switch off" NIL (|TimeOffSet?| T) (|TimeOffSet?| NIL))
("Change time to switch off" |SetOffTime| "Allows you to alter the time when this monitor will
automatically switch off" NIL NIL (|TimeOffSet?| T))) |Off| O\n) ("Choose Path (pair)"
|ChoosePairOfPath| "Choose a pair of path amongst which those processes are of intested to
investigate, then attach the collecter to those zoom nodes." (SUBITEMS ("Choose Several Pairs of
Path" |ChooseMultiPairPath| "Attach the collecter to a selected variable")) |Off| O\n) ("Dettach
Pathes" |DettachVariable| "Attach the collecter to those variables which needs to record the
number of queue in the page when they occurs." (SUBITEMS ("Dettach All Pairs of Path"
|DettachAllVariables| "Attach the collecter to a selected variable")) |Off| O\n) ("Choose Ind.
Variable" |ChooseIndVariable| "Choose an independent variable of which is interested to generate
metamodel for this run." (SUBITEMS ("Choose Ind. Variables" |ChooseMultiIndVariables| "Attach the
collecter to a selected variable")) |Off| O\n) ("Steady-State Indication" |SSIndication|
"Indicate user whether the system has reached steady-state, so that collected data can be adopt
to generate metamodel" (SUBITEMS ("Steady-State Indication(several periods)" |SPIndication|
"Indicate user whether the system has reached steady-state, so that collected data can be adopt
to generate metamodel")) |Off| O\n)) NIL NIL)
                                 ))
   (|InstanceVariables| (|IndVar| 0 |doc|
                                 "Remember which independent variable(s) are going to be monitor")
         (|numberOfQ| 0 |doc| "Number of queue on attached page")
         (|record| (NIL)
                |doc|
      "Association list in which record flow input time with associated number in queue of the
page."
                )
         (|flowList| NIL |doc| "List of flows when it occurs the number in queue will be
recorded.")
         (|boxList| NIL |doc| "List of boxes where the process time will be recorded down.")
         (|typeList| NIL |doc| "Association list of product type - corresponding number")
         (|log| NIL |doc|
                "nested ALst used to store appropriate stream pointers or filenames as
```

```
required")
          (|alreadyReset?| NIL |doc| |flag| |used| |to| |prevent| |self| |from| |reseting| |more|
                |than| |once| |as| \a |result| |of| |activity|)
          (|image-bm| STAGE-BM)))


(|\\BatchMethodDefs|)
(METH |Collecter|     |AttachIdemNode| NIL
      "attach self to all IDEMBOX of an IDEF0 new page." (|category| (MONITOR)))

(METH |Collecter|     |AttachPartialZoomNode| NIL

 "Promt for user to pick up nodes of which the process are of interest, then to ask for select
paths"
      (|category| (|Collecter|)))


(METH |Collecter|     |AttachZoomNode| NIL
      "attach self to all zoom nodes of a transition map." (|category| (|Collecter|)))


(METH |Collecter|     |ChooseAnObj| NIL
      "select an object which can be either a path or a node" (|category| (|Collecter|)))


(METH |Collecter|     |ChooseAnVariable| NIL
      "select a variable, which can be a input flow or a process time, to record the number in
queue when it occurs."
      (|category| (|Collecter|)))


(METH |Collecter|     |ChooseMultiPairPath| NIL
      "To choose more than one pair of path." (|category| (|Collecter|)))

(METH |Collecter|     |ChoosePairOfPath| NIL
      "To choose one of flow threds, or to select part of processes of the only flow thred."
      (|category| (|Collecter|)))


(METH |Collecter|     |ChooseVarMenu| NIL
      "prompt a menu for user to choose an independent variable." (|category| (|Collecter|)))


(METH |Collecter|     |ChooseVariables| NIL
      "Select several variables when the flow occurs, the number of queue is required to record."
      (|category| (|Collecter|)))


(METH |Collecter|     |ConvertTypeToNumber| (|outputDataList|)
      "transfer product type - an atom into a numerical number" (|category| (|Collecter|)))


(METH |Collecter|     |DSPWIPHist| (|WIPdata| |data|)
      "display the WIP histogram" (|category| (|Collecter|)))


(METH |Collecter|     |DettachAllVariables| NIL
      "Dettach all variables been selected before." (|category| (|Collecter|)))


(METH |Collecter|     |DettachVariable| NIL
      "dettach a variable that user selected already." (|category| (|Collecter|)))


(METH |Collecter|     |EventChanged| NIL
      "if event has changed, true - add hook" (|category| (|Collecter|)))


(METH |Collecter|     |ExpungeData| NIL
      "delete all of the file present on the current log." (|category| (|Collecter|)))


(METH |Collecter|     |GetMetaDescription| NIL
      "Prompt for description of file to be saved." (|category| (|Collecter|)))


(METH |Collecter|     |GetMetaIndexFileName| NIL
      "Return index file name in which data file name is saved." (|category| (|Collecter|)))
```

```
(METH |Collecter|     |GetQLength| NIL
      "Collect qLength (in term of number of order waiting in the queue to process) through
simulation controller, the number does not include those order being processing."
      (|category| (|Collecter|)))

(METH |Collecter|     |GetQLength2| NIL
      "Collect number of Working In Process (include those being processing)" (|category| (
|Collecter|
                                                                                          )))

(METH |Collecter|     |GetWIPData| (|data|)
      "snap WIP data off" (|category| (|Collecter|)))

(METH |Collecter|     |InitialLog| NIL
      "To obtain independent variables from user, and then set logger." (|category|
(|Collecter|)))

(METH |Collecter|     |PickPath| (|whichPath|)
      "pick a path, which can be either input or output path." (|category| (|Collecter|)))

(METH |Collecter|     |PickPaths| (|flag|)
      "Select several paths, data is required to collect when flow occurs along the paths."
      (|category| (|Collecter|)))

(METH |Collecter|     |ProcessData| (|input1FlowRecord| |input2FlowRecord| |input3FlowRecord|
|outputFlowRecord| |key|)
      "To process original data to meet metamodel generator's requirement." (|category|
(|Collecter|)
                                                                                          ))

(METH |Collecter|     |ProcessEvent| (|event| |timeChanged?|)
      "Sent to self when an event occurs - take account if it is on the attached page."
(|category|
(|Collecter|
                                                                                          )))

(METH |Collecter|     |PutToLog| (|keys|)
      "finds appropriate fileStream on log." (|category| (|Collecter|)))

(METH |Collecter|     |ReSet| (|timeWarp?|)
      "ensure clear the iv of itself" (|category| (SIMULATION-OBJECT)))

(METH |Collecter|     |Record| (|event| |flag|)
      "To record data required by user. If flag is T, WIP will be recorded, otherwise it takes
record depanding on independent variable selected. "
      (|category| (|Collecter|)))

(METH |Collecter|     |RestData| (|TypeLst| |typeAndBatch|)
      "deal with rest of data" (|category| (|Collecter|)))

(METH |Collecter|     |SPIndication| NIL
      "fetch WIP data been colected and then indicate the system state by analysis the data."
      (|category| (|Collecter|)))

(METH |Collecter|     |SSIndication| NIL
      "fetch WIP data been colected and then indicate the system state by analysis the data."
      (|category| (|Collecter|)))

(METH |Collecter|     |SState?| (|data| |intervals|)
      "analyze histogram of WIP, thus user can determine whether the system has reached steady
state"
      (|category| (|Collecter|)))
```

```
(METH |Collecter|     |SaveData| NIL
      "save data into a file which is with extend." (|category| (|Collecter|)))

(METH |Collecter|     |SaveDataIfWanted| NIL
      "Save off the data if the user confirms that he wants to save." (|category| (|Collecter|)))

(METH |Collecter|     |Start| NIL
      "check initial setting, if T - start, otherwise tell user to select them." (|category|
(MONITOR
                                                                                          )))

(METH |Collecter|     |StoreValue| NIL
      "save data into a file." (|category| (|Collecter|)))

(METH |Collecter|     |SwitchOff| NIL
      "switch off self." (|category| (MULTIPLE-ATTRIBUTE-MONITOR)))

(METH |Collecter|     |SwitchOn| NIL
      "switch self on at first, then attach self to all the nodes of the map" (|category| (

MULTIPLE-ATTRIBUTE-MONITOR
                                                                                          )))

(METH |Collecter|     |TakeRecord| (|event|)
      "record data that user request on self." (|category| (|Collecter|)))

(METH |Collecter|     |\\GetFileStream| (|keys|)
      "Method documentation." (|category| (|Collecter|)))

(METH |Collecter|     |\\GetLogFileName| (|keys|)
      "Method documentation." (|category| (|Collecter|)))


(|Method| (  (|Collecter| |AttachIdemNode|)   |self|)
   "attach self to all IDEMBOX of an IDEF0 new page."
   (LET* ((|drawing| (@ |drawing|))
          (|objLst| (@ |drawing| |objectList|))
          (|boxes| (  |for|   |boxes|   |in|   |objLst|   |when|   (_ |boxes| |InstOf!| 'IDEMBOX)
                      |collect|   |boxes|)))
         (  |if|   |boxes|
             |then|   (LET ((|nodeLst| (LIST NIL)))
                         (  |for|   \i   |from|   1   |to|   (LENGTH |boxes|)
                            |do|   (SETQ |nodeLst| (APPEND |nodeLst|
                                                            (LIST (CAR (NTH |boxes| \i))))))
                         (_@ |attachees| (CDR |nodeLst|))))))

(|Method| (  (|Collecter| |AttachPartialZoomNode|)   |self|)

 "Promt for user to pick up nodes of which the process are of interest, then to ask for select
paths"
   (_ (@ |self| |drawing|)
      |ClearPromptWindow|)
   (LET ((|numOfProcess| (_ (@ |:drawing|)
                            |PromptForWord| "How many nodes are you interested? "))
         (  |for|   \i   |from|   1   |to|   |numOfProcess|
            |do|   (_ |self| |PrintMessage| "Select " \i " node")
                  (_ |self| |ChooseAnObj|))
         (_ |self| |ChoosePairOfPath|)))

(|Method| (  (|Collecter| |AttachZoomNode|)   |self|)
   "attach self to all zoom nodes of a transition map."
   (LET* ((|drawing| (@ |drawing|))
          (|objLst| (@ |drawing| |objectList|))
          (|zoomNodes| (  |for|   |nodes|   |in|   |objLst|   |when|   (OR (_ |nodes| |InstOf!|
                                                                                  ,

ZOOM-STAGE-PROCESS)
                                                                          (_ |nodes| |InstOf!|
```

```
ZOOM-ROUTER-PROCESS)
                                                                                    )   |collect|   |nodes|)))
        (  |if|   |zoomNodes|
           |then|   (LET ((|nodeLst| (LIST NIL)))
                        (  |for|   \i   |from|   1   |to|   (LENGTH |zoomNodes|)
                             |do|   (SETQ |nodeLst| (APPEND |nodeLst|
                                                    (LIST (CAR (NTH |zoomNodes|
\i))))))
                        (_@ |attachees| (CDR |nodeLst|))))))

(|Method| (  (|Collecter| |ChooseAnObj|)   |self|)
   "select an object which can be either a path or a node"
   (LET ((|selectedVar| (_ (@ |:drawing|)
                           |SelectObject|))
         (|page?| (@ |:drawing|)))
       (  |if|   (_ |page?| |InstOf!| 'IDEF0NEWPAGE)
            |then|   (  |if|   (AND |selectedVar| (_ |selectedVar| |InstOf!| 'ARROW))
                        |then|   (  |if|   (@ |:flowList|)
                                     |then|   (_@ |self| |flowList| (APPEND (@ |:flowList|)
                                                                           (LIST
|selectedVar|
                                                                                 )))
                                  |else|   (_@ |self| |flowList| (LIST |selectedVar|)))
                     |elseif|   (AND |selectedVar| (_ |selectedVar| |InstOf!| 'IDEMBOX))
                       |then|   (  |if|   (@ |:boxList|)
                                     |then|   (_@ |self| |boxList| (APPEND (@ |:boxList|)
                                                                          (LIST
|selectedVar|)
                                                                          ))
                                  |else|   (_@ |self| |boxList| (LIST |selectedVar|)))
                     |elseif|   |selectedVar|
                       |then|   (_ (@ |:drawing|)
                                  |PrintMessage|
                                  "You can only choose flow path or function box.")
                     |else|   (_ (@ |:drawing|)
                                 |PrintMessage|
                "You have to choose flow path or function box before you collect the data you
want.")
                        )
            |elseif|   (_ |page?| |InstOf!| 'IDEMTRANSITIONMAP)
              |then|   (  |if|   (AND |selectedVar| (_ |selectedVar| |InstOf!| 'PATH))
                        |then|   (  |if|   (@ |:flowList|)
                                     |then|   (_@ |self| |flowList| (APPEND (@ |:flowList|)
                                                                           (LIST
|selectedVar|
                                                                                 )))
                                  |else|   (_@ |self| |flowList| (LIST |selectedVar|)))
                     |elseif|   (AND |selectedVar| (OR (_ |selectedVar| |InstOf!|
                                                            'ZOOM-STAGE-PROCESS)
                                                         (_ |selectedVar| |InstOf!|
                                                            'ZOOM-ROUTER-PROCESS)))
                        |then|   (  |if|   (@ |:boxList|)
                                     |then|   (_@ |self| |boxList| (APPEND (@ |:boxList|)
                                                                          (LIST
|selectedVar|)
                                                                          ))
                                  |else|   (_@ |self| |boxList| (LIST |selectedVar|)))
                     |elseif|   |selectedVar|
                       |then|   (_ (@ |:drawing|)
                                  |PrintMessage|
                                  "You can only choose flow path or function box.")
                     |else|   (_ (@ |:drawing|)
                                 |PrintMessage|
                "You have to choose flow path or function box before you collect the data you
want.")
                        ))
       (_ (@ |:drawing|)
          |ClearSelection|)))

(|Method| (  (|Collecter| |ChooseAnVariable|)   |self|)
   "select a variable, which can be a input flow or a process time, to record the number in queue
when it occurs."
   (_ (@ |self| |drawing|)
      |ClearPromptWindow|)
   (_ |self| |PrintMessage|
          "Select the Input Flow that you want to record the number of queue in this page.")
   (LET ((|selectedVar| (_ (@ |:drawing|)
```

```
                                |SelectObject|))
          (|page?| (@ |:drawing|)))
        (  |if|    (_ |page?| |InstOf!| 'IDEF0NEWPAGE)
             |then|    (  |if|    (AND |selectedVar| (_ |selectedVar| |InstOf!| 'ARROW))
                            |then|    (  |if|    (@ |:flowList|)
                                          |then|    (_@ |self| |flowList| (APPEND (@ |:flowList|)
                                                                                  (LIST
|selectedVar|
                                                                                      )))
                                       |else|    (_@ |self| |flowList| (LIST |selectedVar|)))
                                |elseif|    (AND |selectedVar| (_ |selectedVar| |InstOf!| 'IDEMBOX))
                                  |then|    (  |if|    (@ |:boxList|)
                                                |then|    (_@ |self| |boxList| (APPEND (@ |:boxList|)
                                                                                  (LIST
|selectedVar|)
                                                                                  ))
                                       |else|    (_@ |self| |boxList| (LIST |selectedVar|)))
                                |elseif|    |selectedVar|
                                  |then|    (_ (@ |:drawing|)
                                               |PrintMessage|
                                               "You can only choose flow path or function box.")
                                |else|    (_ (@ |:drawing|)
                                             |PrintMessage|
                   "You have to choose flow path or function box before you collect the data you
want.")
                                )
             |elseif|    (_ |page?| |InstOf!| 'IDEMTRANSITIONMAP)
               |then|    (  |if|    (AND |selectedVar| (_ |selectedVar| |InstOf!| 'PATH))
                            |then|    (  |if|    (@ |:flowList|)
                                          |then|    (_@ |self| |flowList| (APPEND (@ |:flowList|)
                                                                                  (LIST
|selectedVar|
                                                                                      )))
                                       |else|    (_@ |self| |flowList| (LIST |selectedVar|)))
                                |elseif|    (AND |selectedVar| (OR (_ |selectedVar| |InstOf!|
                                                                        'STEM-STAGE-PROCESS)
                                                                    (_ |selectedVar| |InstOf!|
                                                                        'STEM-ROUTER-PROCESS)))
                                  |then|    (  |if|    (@ |:boxList|)
                                                |then|    (_@ |self| |boxList| (APPEND (@ |:boxList|)
                                                                                  (LIST
|selectedVar|)
                                                                                  ))
                                       |else|    (_@ |self| |boxList| (LIST |selectedVar|)))
                                |elseif|    |selectedVar|
                                  |then|    (_ (@ |:drawing|)
                                               |PrintMessage|
                                               "You can only choose flow path or function box.")
                                |else|    (_ (@ |:drawing|)
                                             |PrintMessage|
                   "You have to choose flow path or function box before you collect the data you
want.")
                                ))
        (_ (@ |:drawing|)
           |ClearSelection|)))

(|Method| (  (|Collecter| |ChooseMultiPairPath|)   |self|) "To choose more than one pair of path."
   (_ (@ |self| |drawing|)
      |ClearPromptWindow|)
   (  |if|    (@ |:flowList|)
        |then|    (  |if|    (MOUSECONFIRM
            "Some path has already been selected, do you sure that you want to change or to add
on? "
                                NIL (_ |self| |\\GetBestMsgStream|))
                      |then|

                            (* |;;| "to be sure that user want to change")

                           (LET ((|theLst| (@ |:flowList|))
                                 |names| |num|)
                                (  |for|    \i    |from|    1    |to|    (LENGTH (@ |:flowList|))
                                     |do|    (SETQ |names|
                                                (APPEND |names| (LIST (@ (CAR (NTH |theLst| \i))
                                                                         |name|)))))
                                (  |if|    (MOUSECONFIRM (CONCAT "Paths (input and output) named "
                                                                 |names|
 " has already been chosen. Do you want to add more pair of paths on or to set new pairs of
paths? "
                                                                 )
                                                   NIL
```

```
                (_ |self| |\\GetBestMsgStream|))
|then|

      (* |;;| "adding new pair of paths on existing pairs.")

    (SETQ |num| (_ (@ |:drawing|)
                   |PromptForWord|
       "Entre the number of pair of path you want to add
on: ")
                )
                ( |for|   \i   |from|   1   |to|   |num|
                    |do|   (_ |self| |PickPaths|))
                |else|

                (* |;;| "prompt user for re-set multi pair of path")

                (_@ |:flowList| NIL)
                (_ |self| |ChooseMultiPairPath|))))
      |else|

            (* |;;| "select multi pair of paths initial")

          (LET ((|num| (_ (@ |:drawing|)
                          |PromptForWord|
                          "Entre the number of information threds you want to monitor: ")))
               ( |for|   \i   |from|   1   |to|   |num|   |do|   (_ |self| |PickPaths|)))))

(|Method| ( (|Collecter| |ChoosePairOfPath|)   |self|)
   "To choose one of flow threds, or to select part of processes of the only flow thred."
   (_ (@ |self| |drawing|)
      |ClearPromptWindow|)
   ( |if|   (@ |:flowList|)
       |then|

            (* |;;| "paths has been selected, prompt for re-selection.")

          ( |if|   (MOUSECONFIRM (CONCAT (@ (CAR (@ |:flowList|))
                                            |name|)
                                         " "
                                         (@ (CADR (@ |:flowList|))
                                            |name|)

        " as input and output paths has already been chosen. Do you want to set new pair of
paths? "
                                         )
                                NIL
                                (_ |self| |\\GetBestMsgStream|))
                    |then|   (_@ |:flowList| NIL)
                        (_ |self| |PickPaths|))
      |else|   (_ |self| |PickPaths|)))

(|Method| ( (|Collecter| |ChooseVarMenu|)   |self|)
   "prompt a menu for user to choose an independent variable."
   (LET ((|menuItem| ( |create|   MENU
                       ITEMS _ '(("WIP" (_@ |:IndVar| 1))
                                 ("WIP and Product Type (PT)" (_@ |:IndVar| 2))
                                 ("WIP and Order Batch Size (OBS)" (_@ |:IndVar| 3))
                                 ("WIP, PT and OBS" (_@ |:IndVar| 4)))
                       TITLE _ "WHICH VARIABLE")))
       ( |if|   |menuItem|
           |then|   (MENU |menuItem|))))

(|Method| ( (|Collecter| |ChooseVariables|)   |self|)
   "Select several variables when the flow occurs, the number of queue is required to record."
   (_ (@ |self| |drawing|)
      |ClearPromptWindow|)
   (LET ((|num| (_ (@ |:drawing|)
                   |PromptForWord| "Entre the number of information threds you want to monitor:
")))
       ( |for|   \i   |from|   1   |to|   |num|   |do|   (_ |self| |ChooseAnVariable|))))

(|Method| ( (|Collecter| |ConvertTypeToNumber|)   |self| |outputDataList|)
   "transfer product type - an atom into a numerical number"
   (LET ((|TLst| (LIST NIL))
         (|referLst| (LIST NIL))
         (\n 1))
       ( |for|   \i   |from|   1   |to|   (LENGTH |outputDataList|)
           |do|   (LET ((|type| (CADDR (CAR (NTH |outputDataList| \i)))))
                       ( |if|   (NOT (MEMB |type| |referLst|))
                            |then|   (SETQ |referLst| (APPEND |referLst| (LIST |type|)))
```

```
                                        (PUTASSOC |type| \n |TLst|)
                                        (SETQ \n (ADD1 \n)))))
        (CDR |TLst|)))

(|Method| (  (|Collecter| |DSPWIPHist|)   |self| |WIPdata| |data|) "display the WIP histogram"
   (LET ((|window| (CREATEW (GETREGION 8 5)
                            "Display Window"))
         (|tempList| NIL)
         (|gap| (FIX (QUOTIENT (LENGTH |data|)
                               (LENGTH |WIPdata|))))
         |smallest| |largest| |value| |fromTime| |toTime|)
        (  |for|   \i   |from|   1   |to|   (LENGTH |data|)
           |do|   (SETQ |tempList| (APPEND |tempList| (LIST (CAAR (NTH |data| \i))))))
        (SETQ |smallestNum| (SMALLEST.IN.LIST |tempList|))
        (SETQ |largestNum| (LARGEST.IN.LIST |tempList|))
        (  |for|   \i   |from|   1   |to|   (LENGTH |WIPdata|)
           |do|   (TERPRI |window|)
                 (TERPRI |window|)
                 (PRINTOUT |window| .SP 5 "WIP" .SP 5 "Frequency")
                 (TERPRI |window|)
                 (TERPRI |window|)
                 (LET ((|partList| (CAR (NTH |WIPdata| \i)))
                       (\n |smallestNum|))
                      (SETQ |fromTime| (CADAR (NTH |data| (IPLUS 1 (TIMES |gap| (IDIFFERENCE \i
1))))
                                                ))
                      (SETQ |toTime| (CADAR (NTH |data| (TIMES |gap| \i))))
                      (  |for|   \j   |from|   |smallestNum|   |to|   |largestNum|
                         |do|   (  |if|   (ASSOC \j |partList|)
                                    |then|   (SETQ |value| (CDR (ASSOC \j |partList|)))
                                    |else|   (SETQ |value| 0))
                              (PRINTOUT |window| .SP 5 \n .SP 9 |value|)
                              (TERPRI |window|)
                              (SETQ \n (ADD1 \n))))
                 (TERPRI |window|)
                 (PRINTOUT |window| .SP 5 "From time " |fromTime| "  -- " |toTime|))))

(|Method| (  (|Collecter| |DettachAllVariables|)   |self|)
   "Dettach all variables been selected before."
   (_@ |:flowList| NIL)
   (_@ |:boxList| NIL))

(|Method| (  (|Collecter| |DettachVariable|)   |self|)
   "dettach a variable that user selected already."
   (LET* ((|flows| (@ |:flowList|))
          (|boxes| (@ |:boxList|))
          (|flowsName| (  |for|   \i   |from|   1   |to|   (LENGTH |flows|)
                          |collect|   (@ (CAR (NTH |flows| \i))
                                        |name|)))
          (|boxesName| (  |for|   \i   |from|   1   |to|   (LENGTH |boxes|)
                          |collect|   (@ (CAR (NTH |boxes| \i))
                                        |contents|)))
          |items| \j |remover| |which|)
         (  |if|   (AND |flowsName| |boxesName|)
            |then|   (SETQ |items| (LIST |flowsName| |boxesName|))
          |elseif|   |flowsName|
            |then|   (SETQ |items| |flowsName|)
          |elseif|   |boxesName|
            |then|   (SETQ |items| |boxesName|)
         (  |if|   |items|
            |then|   (SETQ |which| (MENU (  |create|   MENU
                                             ITEMS _ |items|
                                             TITLE _ "Which variable?"))))
         (  |if|   (MEMBER |which| |flowsName|)
            |then|   (SETQ \j (CAR (  |for|   \i   |from|   1   |to|   (LENGTH |flows|)
                                     |collect|   \i
                                     |when|   (EQ |which| (@ (CAR (NTH |flows| \i))
                                                              |name|)))))
                     (_@ |flowList| (REMOVE (CAR (NTH |flows| \j))
                                             |flows|))
          |elseif|   (MEMBER |which| |boxesName|)
            |then|   (SETQ \j (CAR (  |for|   \i   |from|   1   |to|   (LENGTH |boxes|)
                                     |collect|   \i
                                     |when|   (EQ |which| (@ (CAR (NTH |boxes| \i))
                                                              |contents|)))))
                     (_@ |boxList| (REMOVE (CAR (NTH |boxes| \j))
                                            |boxes|)))))

(|Method| (  (|Collecter| |EventChanged|)   |self|) "if event has changed, true - add hook"
   (  |if|   (@ |:state:,monitoringEvents?|)
```

```
        |then|   (_ |self| |PrintMessage| (CONCAT "Events were already being monitored by  "
                                                 (OR (@ |:name|)
                                                     (|ClassName| |self|))))
      |else|   (_ |self| |ChangeState| '|monitoringEvents?| T)
          (_ (@ T:drawing:controller|)
             |AddEventHook| |self|)))

(|Method| (  (|Collecter| |ExpungeData|)   |self|)
   "delete all of the file present on the current log."
   (|OperateOnLogFiles| (@ |:log|)
         'DELFILE))

(|Method| (  (|Collecter| |GetMetaDescription|)   |self|)
   "Prompt for description of file to be saved."
   (RESETLST
       (RESETSAVE (TTY.PROCESS (THIS.PROCESS)))
       (RESETSAVE (TTYDISPLAYSTREAM (  |if|   (  |type?|   |instance| (@ |:drawing|))
                                       |then|   (_ (@ |:drawing|)
                                                   |GetPromptWindow|))))
       (LET ((|description| (_ |self| |PromptForAtom|
                                "Enter a brief description of the logged data: ")))
            (  |if|   (NULL |description|)
               |then|   (_ |self| |PrintMessage| "Using default description")
                   (SETQ |description| (CONCAT (@ |:controller:thisID|)
                                               "data recorded on" NIL
                                               (GDATE NIL (DATEFORMAT NO.LEADING.SPACES SPACES
                                                                      NO.SECONDS DAY.OF.WEEK))
                                               "by " INITIALS)))
            (MKSTRING |description|))))

(|Method| (  (|Collecter| |GetMetaIndexFileName|)   |self|)
   "Return index file name in which data file name is saved."
   (PACKFILENAME 'NAME (@ |:controller:thisID|)
         'EXTENSION
         'IDEM-META-INDEX))

(|Method| (  (|Collecter| |GetQLength|)   |self|)
   "Collect qLength (in term of number of order waiting in the queue to process) through
simulation controller, the number does not include those order being processing."
   (LET* ((|controller| (@ |controller|))
          (|queue| (@ |controller| |queue|))
          (|attachee| (@ |:attachees|))
          (|numberQ| 0))
         (  |if|   |queue|
            |then|   (  |for|   \i   |from|   1   |to|   (LENGTH |queue|)
                       |do|   (LET ((|qLocation| (@ (CAR (NTH |queue| \i))
                                                      |whom|)))
                                  (  |if|   (AND (MEMBER |qLocation| |attachee|))
                                     |then|   (SETQ |numberQ| (ADD1 |numberQ|))
                                         (_@ |numberOfQ| |numberQ|)))))))

(|Method| (  (|Collecter| |GetQLength2|)   |self|)
   "Collect number of Working In Process (include those being processing)"
   (LET ((|attachee| (@ |:attachees|))
         (|numberQ| 0))
        (  |for|   \i   |from|   1   |to|   (LENGTH |attachee|)
           |do|   (SETQ |numberQ| (IPLUS |numberQ| (@ (CAR (NTH |attachee| \i))
                                                       |localQLength|))))
        (_@ |numberOfQ| |numberQ|)))

(|Method| (  (|Collecter| |GetWIPData|)   |self| |data|) "snap WIP data off"
   (PROG ((|pureData| (CDR |data|))
          (|WIPandTime| NIL))
         (  |for|   \i   |from|   1   |to|   (LENGTH |pureData|)
            |do|   (SETQ |WIPandTime| (APPEND |WIPandTime| (LIST (CDAR (NTH |pureData| \i))))))
         (RETURN |WIPandTime|)))

(|Method| (  (|Collecter| |InitialLog|)   |self|)
   "To obtain independent variables from user, and then set logger."
   (LET ((|whichOne| (_ |self| |ChooseVarMenu|))
         |key|)
        (  |if|   (EQ |whichOne| 1)
           |then|   (SETQ |key| 'WIP)
           |elseif|   (EQ |whichOne| 2)
           |then|   (SETQ |key| '(WIP |Product|))
           |elseif|   (EQ |whichOne| 3)
           |then|   (SETQ |key| '(WIP |OrderBatchSize|))
           |elseif|   (EQ |whichOne| 4)
           |then|   (SETQ |key| '(WIP |Product| |OrderBatchSize|)))
        (_ |self| |PutToLog| (LIST |key|))))
```

```
(|Method| (  (|Collecter| |PickPath|)   |self| |whichPath|)
   "pick a path, which can be either input or output path."
   (_ (@ |self| |drawing|)
      |ClearPromptWindow|)
   (  |if|    (EQ |whichPath| 1)
         |then|    (_ |self| |PrintMessage| "Select the Input Flow Path.")
       |elseif|    (EQ |whichPath| 2)
         |then|    (_ |self| |PrintMessage| "Select the output Flow Path."))
   (_ |self| |ChooseAnObj|)
   (_ (@ |self| |drawing|)
      |ClearPromptWindow|))

(|Method| (  (|Collecter| |PickPaths|)   |self| |flag|)
   "Select several paths, data is required to collect when flow occurs along the paths."
   (_ (@ |self| |drawing|)
      |ClearPromptWindow|)
   (  |for|   \i   |from|   1   |to|   2   |do|   (_ |self| |PickPath| \i)))

(|Method| (  (|Collecter| |ProcessData|)   |self| |input1FlowRecord| |input2FlowRecord|
                                                  |input3FlowRecord| |outputFlowRecord| |key|)
   "To process original data to meet metamodel generator's requirement."
   (PROG ((|outputDataList| (CDR |outputFlowRecord|))
          |orderNumber| |inputDL1| |inputDL2| |wip1| |wip2| |inputTime1| |inputTime2|
|outputTime|
          |newRecord| |processTime| |TypeLst| |others|)
         (  |for|   \i   |from|   1   |to|   (LENGTH |outputDataList|)
            |do|   (  |if|   (AND (NULL |input2FlowRecord|)
                                 (NULL |input3FlowRecord|))
                        |then|                                  (* \; "only one input path")
                            (SETQ |orderNumber| (CAAR (NTH |outputDataList| \i)))
                            (SETQ |inputDL1| (CDR |input1FlowRecord|))
                            (SETQ |wip1| (CADAR (NTH |inputDL1| \i)))
                            (SETQ |inputTime1| (CADDR (CAR (NTH |inputDL1| \i))))
                            (SETQ |outputTime| (CADR (CAR (NTH |outputDataList| \i))))
                                                                  (* \; "get process time")
                            (SETQ |processTime| (DIFFERENCE |outputTime| |inputTime1|))
                            (  |if|   (AND (LISTP |key|)
                                           (MEMB '|Product| |key|))
                                  |then|                          (* \;
    "automatically to form a product type list in which each type has a corresponding numerical
number")
                                      (SETQ |TypeLst| (_ |self| |ConvertTypeToNumber|
                                                          |outputDataList|))
                                      (_@ |:typeList| |TypeLst|)
                                                                  (* \;
                                                        "convert product type into a usable
number")
                                      (SETQ |others| (_ |self| |RestData| |TypeLst|
                                                         (CDDR (CAR (NTH |outputDataList|
\i)
                                                                        ))))
                                                                  (* \;
                                      "investigation is one of WIP-Product or
WIP-Product-BatchSize")
                                      (SETQ |newRecord| (APPEND |newRecord|
                                                                (LIST (APPEND (LIST
|orderNumber|
                                                                               |wip1|
|inputTime1|)
                                                                              |others|
                                                                              (LIST
|processTime|)))
                                                                ))
                                 |elseif|   (AND (LISTP |key|)
                                                 (MEMB '|OrderBatchSize| |key|))
                                   |then|   (SETQ |others| (CDDR (CAR (NTH |outputDataList| \i))))
                                                                  (* \; "investigation is
WIP-BatchSize")
                                      (SETQ |newRecord| (APPEND |newRecord|
                                                                (LIST (APPEND (LIST
|orderNumber|
                                                                               |wip1|
|inputTime1|)
                                                                              |others|
                                                                              (LIST
|processTime|)))
                                                                ))
```

```
                                    |elseif|   (NOT (LISTP |key|))
                                      |then|                        (* \; "investigation is WIP only")
                                          (SETQ |newRecord| (APPEND |newRecord|
                                                                    (LIST (LIST |orderNumber|
|wip1|
                                                                                |inputTime1|
                                                                                |processTime|)))))
                              |elseif|   (NULL |input3FlowRecord|)
                              |then|                              (* \; "there are two input paths")
                                  (SETQ |orderNumber| (CAAR (NTH |outputDataList| \i)))
                                  (SETQ |inputDL1| (CDR |input1FlowRecord|))
                                  (SETQ |wip1| (CADAR (NTH |inputDL1| \i)))
                                  (SETQ |inputTime1| (CADDR (CAR (NTH |inputDL1| \i))))
                                  (SETQ |inputDL2| (CDR |input2FlowRecord|))
                                  (SETQ |wip2| (CADAR (NTH |inputDL2| \i)))
                                  (SETQ |inputTime2| (CADDR (CAR (NTH |inputDL2| \i))))
                                  (SETQ |outputTime| (CADR (CAR (NTH |outputDataList| \i))))
                                  (  |if|   (AND (LISTP |key|)
                                                 (MEMB '|Product| |key|))
                                       |then|   (SETQ |TypeLst| (_ |self| |ConvertTypeToNumber|
                                                                          |outputDataList|))
                                           (_@ |:typeList| |TypeLst|)
                                                                  (* \;
                                                                 "convert product type into a usable
number")
                                           (SETQ |others| (_ |self| |RestData| |TypeLst|
                                                                  (CDDR (CAR (NTH |outputDataList|
\i)
                                                                             ))))
                                                                  (* \;
                                           "investigation is one of WIP-Product or
WIP-Product-BatchSize")
                                           (SETQ |newRecord|
                                            (APPEND |newRecord|
                                                    (LIST (APPEND (LIST |orderNumber| |wip1| |wip2|
                                                                        |inputTime1| |inputTime2|)
                                                                  |others|
                                                                  (LIST |outputTime|)))))
                                    |elseif|   (AND (LISTP |key|)
                                                     (MEMB '|OrderBatchSize| |key|))
                                      |then|                      (* \; "investigation is WIP-BatchSize")
                                          (SETQ |others| (CDDR (CAR (NTH |outputDataList| \i))))
                                          (SETQ |newRecord|
                                           (APPEND |newRecord|
                                                   (LIST (APPEND (LIST |orderNumber| |wip1| |wip2|
                                                                       |inputTime1| |inputTime2|)
                                                                 |others|
                                                                 (LIST |outputTime|)))))
                                    |elseif|   (NOT (LISTP |key|))
                                      |then|                      (* \; "investigation is WIP only")
                                          (SETQ |newRecord|
                                           (APPEND |newRecord|
                                                   (LIST (LIST |orderNumber| |wip1| |wip2|
|inputTime1|
                                                                 |inputTime2| |outputTime|)))))
                              |else|                                (* \; "there are three input paths")
                                  (CL:BREAK)))
        (RETURN |newRecord|)))

(|Method| (  (|Collecter| |ProcessEvent|)   |self| |event| |timeChanged?|)
   "Sent to self when an event occurs - take account if it is on the attached page."
   (LET ((|page?| (@ |:drawing|))
         (|monitoredEvents| (  |for|   \i   |from|   1   |to|   (LENGTH (@ |:flowList|))
                                 |collect|   (@ (CAR (NTH (@ |:flowList|)
                                                          \i))
                                                 |name|)))
         (|eventPath| (@ (@ (@ |event| |package|)
                            |datum|)
                         |destination|))
         (|attachee| (@ |:attachees|))
         (|selector| (@ |event| |selector|)))
        (  |if|   (_ |page?| |InstOf!| 'IDEF0NEWPAGE)
             |then|   (  |if|   (AND (MEMBER |eventPath| |monitoredEvents|)
                                      (EQ |selector| '|Arrives|))
                            |then|   (_ |self| |GetQLength2| |event|)
                                     (_ |self| |TakeRecord| |event|))
           |elseif|   (_ |page?| |InstOf!| 'IDEMTRANSITIONMAP)
             |then|   (  |if|   (AND (|StringEQAtom| |eventPath| (CAR (LAST |monitoredEvents|)))
                                      (EQ |selector| '|Arrives|))
                            |then|                                (* \;
```

```
                    "to record times and value of other independent variable - if there is
any")
                         (_ |self| |Record| |event| NIL)
                    |else|   (  |for|   \i   |from|   1   |to|   (IDIFFERENCE (LENGTH
|attachee|
                                                                                  )
                                                                          1)
                              |do|   (  |if|   (AND (|StringEQAtom| |eventPath|
                                                        (CAR (NTH |monitoredEvents|
\i)
                                                                        ))
                                                              (EQ |selector| '|Arrives|))
                                                        |then|
                                                              (* \;
                         "to record WIP of the system when a flow occurs along the selected input
path")
                                                        (_ |self| |GetQLength2| |event|)
                                                        (_ |self| |Record| |event| T))))
                 |else|   (ERROR "attached to which page?")))
   |event|)

(|Method| (  (|Collecter| |PutToLog|)   |self| |keys|) "finds appropriate fileStream on log."
   (LET ((|fileStream| (_ |self| |\\GetFileStream| |keys|)))
        (PRINT |keys| |fileStream|)
        |fileStream|))

(|Method| (  (|Collecter| |ReSet|)   |self| |timeWarp?|) "ensure clear the iv of itself"
   (  |if|   (EQ (@ |:state|)
                 'O\n)
        |then|   (  |if|   (AND (@ |:log|)
                                 (NULL (@ |:alreadyReset?|)))
                      |then|                                   (* \;
                                "there is logged data. Prompt for user to comfirm to save the
data.")
                                (_ |self| |StoreValue|)
                                (_ |self| |SaveDataIfWanted|)
                                (_@ |:log| NIL)
                                (_@ |:alreadyReset?| T))            (* \; "reset self's IV")
             (_@ |:numberOfQ| 0)
             (_@ |:record| (LIST NIL))
             (_@ |:typeList| NIL)
             (_ (@ |:drawing|)
                |ClearPromptWindow|)
             (  |if|   (MOUSECONFIRM
                                "Do you want to re-select nodes for next run or remain as current
run?"
                                NIL (_ |self| |\\GetBestMsgStream|))
                    |then|   (_@ |:flowList| NIL)
                        (_@ |:boxList| NIL)))
   (|_Super| |self| |ReSet| |timeWarp?|))

(|Method| (  (|Collecter| |Record|)   |self| |event| |flag|)
   "To record data required by user. If flag is T, WIP will be recorded, otherwise it takes
record depanding on independent variable selected. "
   (LET*
    ((|controller| (@ |controller|))
     (|time| (@ |controller| |timeNow|))
     (|ref| (@ (@ (@ |event| |package|)
                  |datum|)
               |reference|))
     (|destination| (@ (@ (@ |event| |package|)
                          |datum|)
                       |destination|))
     (|pathNames| NIL)
     (|recordLst| (@ |:record|))
     (|whatToRecord| (@ |:IndVar|))
     |pathName|)
    (  |for|   \i   |from|   1   |to|   (LENGTH (@ |:flowList|))
       |do|   (SETQ |pathName| (@ (CAR (NTH (@ |:flowList|)
                                           \i))
                                   |name|))
             (  |if|   |pathNames|
                    |then|   (SETQ |pathNames| (APPEND |pathNames| (LIST |pathName|)))
                 |else|   (SETQ |pathNames| (LIST |pathName|)))))
    (  |if|   |flag|
       |then|                                                   (* \;
                                                        "to record WIP (flow occurs on input path
selected)")
             (  |if|   (_ |page?| |InstOf!| 'IDEFONEWPAGE)
```

```
                    |then|                                        (* \;
                                                                  "collecter is alocated on
IDEFONEWPAGE")
                        (  |if|   (MEMBER |destination| |pathNames|)
                             |then|   (  |if|   (AND (NULL (CAR (@ |:record|)))
                                                      (NULL (ASSOC |destination| |recordLst|)))
                                         |then|   (PUTASSOC |destination|
                                                          (LIST (LIST |ref| (@ |:numberOfQ|)
                                                                      |time|))
                                                          |recordLst|)
                                                 (_@ |:record| (CDR |recordLst|))
                                       |elseif|   (NULL (ASSOC |destination| |recordLst|))
                                         |then|   (PUTASSOC |destination|
                                                          (LIST (LIST |ref| (@ |:numberOfQ|)
                                                                      |time|))
                                                          |recordLst|)
                                                 (_@ |:record| |recordLst|)
                                       |else|   (PUTASSOC |destination|
                                                        (APPEND (CDR (ASSOC |destination|
                                                                            |recordLst|))
                                                                (LIST (LIST |ref| (@
|:numberOfQ|)
                                                                            |time|)))
                                                        |recordLst|)
                                           (_@ |:record| |recordLst|)))
                    |else|                                        (* \;
                                                                  "collecter is allocated on
TRANSITIONMAP")
                        (  |for|   \i   |from|   1   |to|   (LENGTH |pathNames|)
                             |when|   (|StringEQAtom| |destination| (CAR (NTH |pathNames| \i)))
                             |do|   (  |if|   (AND (NULL (CAR (@ |:record|)))
                                                    (NULL (ASSOC |destination| |recordLst|)))
                                        |then|   (PUTASSOC |destination| (LIST (LIST |ref|
                                                                                    (@
|:numberOfQ|)
                                                                                    |time|))
                                                                |recordLst|)
                                                 (_@ |:record| (CDR |recordLst|))
                                       |elseif|   (NULL (ASSOC |destination| |recordLst|))
                                        |then|   (PUTASSOC |destination| (LIST (LIST |ref|
                                                                                    (@
|:numberOfQ|)
                                                                                    |time|))
                                                                |recordLst|)
                                                 (_@ |:record| |recordLst|)
                                       |else|   (PUTASSOC |destination|
                                                        (APPEND (CDR (ASSOC |destination|
|recordLst|))
                                                                (LIST (LIST |ref| (@ |:numberOfQ|)
                                                                            |time|)))
                                                        |recordLst|)
                                           (_@ |:record| |recordLst|))))
                    |else|                                        (* \;
                                                                  "to record data that user wishes (flow occurs on output path
selected)")
                    (LET ((|finishTime| (@ |event| |qEntryTime|))
                          (|productType| (CAR (@ (@ (@ |event| |package|)
                                                    |datum|)
                                                 |data|)))
                          (|orderBatchSize| (CADR (@ (@ (@ |event| |package|)
                                                        |datum|)
                                                     |data|)))
                          (  |if|   (AND (NULL |flag|)
                                         (EQ |whatToRecord| 1))
                             |then|                                       (* \; "to record finish time only")
                                 (  |if|   (EVENP (LENGTH |pathNames|))
                                      |then|   (  |for|   \i   |from|   0   |to|   (LENGTH |pathNames|)
                                                    |by|   2   |when|   (|StringEQAtom| |destination|
                                                                          (CAR (NTH |pathNames| \i)))
                                                    |do|   (  |if|   (AND (NULL (CAR (@ |:record|)))
                                                                           (NULL (ASSOC |destination|
|recordLst|
                                                                                        )))
                                                                |then|   (PUTASSOC |destination|
                                                                                (LIST (LIST |ref|
|finishTime|)
                                                                                      )
                                                                                |recordLst|)
                                                                      (_@ |:record| (CDR |recordLst|))
                                                                |elseif|   (NULL (ASSOC |destination| |recordLst|
```

```
                                                                                    ))
                                                                |then|    (PUTASSOC |destination|
                                                                                 (LIST (LIST |ref|
|finishTime|))
                                                                                    )
                                                                                 |recordLst|))
                                                                        (_@ |:record| |recordLst|))
                                                                |else|    (PUTASSOC |destination|
                                                                                 (APPEND (CDR (ASSOC
|destination|
|recordLst|))
                                                                                      (LIST (LIST |ref|
|finishTime|))
                                                                                      )
                                                                                 |recordLst|))
                                                                        (_@ |:record| |recordLst|)))
                                |else|                                        (* \; "two input pathes and one output
path has been attached, and output path is last in the pathNames list. To record finish time")
                                ( |if|    (AND (NULL (CAR (@ |:record|)))
                                                 (NULL (ASSOC |destination| |recordLst|)))
                                   |then|    (PUTASSOC |destination| (LIST (LIST |ref| |finishTime|))
                                                  |recordLst|)
                                        (_@ |:record| (CDR |recordLst|))
                                |elseif|    (NULL (ASSOC |destination| |recordLst|))
                                   |then|    (PUTASSOC |destination| (LIST (LIST |ref| |finishTime|))
                                                  |recordLst|)
                                        (_@ |:record| |recordLst|)
                                |else|    (PUTASSOC |destination| (APPEND (CDR (ASSOC |destination|
|recordLst|))
                                                                      (LIST (LIST |ref|
|finishTime|)))
                                                  |recordLst|)
                                        (_@ |:record| |recordLst|)))
                        |elseif|    (AND (NULL |flag|)
                                           (EQ |whatToRecord| 2))
                          |then|                                          (* \;
                                                                          "to record finish time and product
type")
                                ( |if|    (EVENP (LENGTH |pathNames|))
                                     |then|    ( |for|    \i    |from|    0    |to|    (LENGTH |pathNames|)
                                                  |by|    2    |when|    (|StringEQAtom| |destination|
                                                                                 (CAR (NTH |pathNames| \i)))
                                                  |do|    ( |if|    (AND (NULL (CAR (@ |:record|)))
                                                                          (NULL (ASSOC |destination|
|recordLst|
                                                                                    )))
                                                                |then|    (PUTASSOC |destination|
                                                                                 (LIST (LIST |ref|
|finishTime|
                                                                                      |productType|))
                                                                                 |recordLst|)
                                                                        (_@ |:record| (CDR |recordLst|))
                                                                |elseif|    (NULL (ASSOC |destination| |recordLst|
                                                                                    ))
                                                                |then|    (PUTASSOC |destination|
                                                                                 (LIST (LIST |ref|
|finishTime|
                                                                                      |productType|))
                                                                                 |recordLst|)
                                                                        (_@ |:record| |recordLst|)
                                                                |else|    (PUTASSOC |destination|
                                                                                 (APPEND (CDR (ASSOC
|destination|
|recordLst|))
                                                                                      (LIST (LIST |ref|
|finishTime|
|productType|)
                                                                                      ))
                                                                                 |recordLst|)
                                                                        (_@ |:record| |recordLst|)))
                                |else|    ( |if|    (AND (NULL (CAR (@ |:record|)))
                                                        (NULL (ASSOC |destination| |recordLst|)))
                                                   |then|    (PUTASSOC |destination| (LIST (LIST |ref|
```

```
                                                                                |finishTime|
|productType|))
                                                |recordLst|)
                                        (_@ |:record| (CDR |recordLst|))
                                  |elseif|   (NULL (ASSOC |destination| |recordLst|))
                                    |then|   (PUTASSOC |destination| (LIST (LIST |ref|
                                                                                |finishTime|
|productType|))
                                                |recordLst|)
                                        (_@ |:record| |recordLst|)
                                  |else|   (PUTASSOC |destination|
                                                (APPEND (CDR (ASSOC |destination|
|recordLst|))
                                                      (LIST (LIST |ref| |finishTime|
                                                                |productType|)))
                                                |recordLst|)
                                        (_@ |:record| |recordLst|)))
                |elseif|   (AND (NULL |flag|)
                                (EQ |whatToRecord| 3))
                  |then|                                  (* \;
                                                "to record finish time and order batch
size")
                     (  |if|   (EVENP (LENGTH |pathNames|))
                          |then|   (  |for|   \i   |from|   0   |to|   (LENGTH |pathNames|)
                                     |by|   2   |when|   (|StringEQAtom| |destination|
                                                            (CAR (NTH |pathNames| \i)))
                                     |do|   (  |if|   (AND (NULL (CAR (@ |:record|)))
                                                           (NULL (ASSOC |destination|
|recordLst|
                                                                        )))
                                                |then|   (PUTASSOC |destination|
                                                                (LIST (LIST |ref|
|finishTime|
|orderBatchSize|))
                                                                |recordLst|)
                                                      (_@ |:record| (CDR |recordLst|))
                                                |elseif|   (NULL (ASSOC |destination| |recordLst|
                                                                        ))
                                                  |then|   (PUTASSOC |destination|
                                                                (LIST (LIST |ref|
|finishTime|
|orderBatchSize|))
                                                                |recordLst|)
                                                      (_@ |:record| |recordLst|)
                                                |else|   (PUTASSOC |destination|
                                                                (APPEND (CDR (ASSOC
|destination|
|recordLst|))
                                                                      (LIST (LIST |ref|
|finishTime|
|orderBatchSize|
                                                                                  )))
                                                                |recordLst|)
                                                      (_@ |:record| |recordLst|)))
                          |else|   (  |if|   (AND (NULL (CAR (@ |:record|)))
                                                  (NULL (ASSOC |destination| |recordLst|)))
                                    |then|   (PUTASSOC |destination| (LIST (LIST |ref|
                                                                                |finishTime|
|orderBatchSize|
                                                                                  ))
                                                |recordLst|)
                                        (_@ |:record| (CDR |recordLst|))
                                  |elseif|   (NULL (ASSOC |destination| |recordLst|))
                                    |then|   (PUTASSOC |destination| (LIST (LIST |ref|
                                                                                |finishTime|
|orderBatchSize|
                                                                                  ))
                                                |recordLst|)
                                        (_@ |:record| |recordLst|)
                                  |else|   (PUTASSOC |destination|
                                                (APPEND (CDR (ASSOC |destination|
```

```
|recordLst|))
                                                                    (LIST (LIST |ref| |finishTime|
                                                                                |orderBatchSize|)))
                                                          |recordLst|)
                                                  (_@ |:record| |recordLst|)))
                  |elseif|   (AND (NULL |flag|)
                                  (EQ |whatToRecord| 4))
                    |then|                                        (* \;
                                          "to record finish time, product type and order batch
size")
                        (  |if|   (EVENP (LENGTH |pathNames|))
                              |then|   (  |for|   \i   |from|   0   |to|   (LENGTH |pathNames|)
                                          |by|   2   |when|   (|StringEQAtom| |destination|
                                                                  (CAR (NTH |pathNames| \i)))
                                          |do|   (  |if|   (AND (NULL (CAR (@ |:record|)))
                                                                (NULL (ASSOC |destination|
|recordLst|
                                                                          )))
                                                          |then|   (PUTASSOC |destination|
                                                                          (LIST (LIST |ref|
|finishTime|
                                                                                      |productType|

|orderBatchSize|))
                                                                          |recordLst|)
                                                                  (_@ |:record| (CDR |recordLst|))
                                                          |elseif|   (NULL (ASSOC |destination| |recordLst|
                                                                          ))
                                                          |then|   (PUTASSOC |destination|
                                                                          (LIST (LIST |ref|
|finishTime|
                                                                                      |productType|

|orderBatchSize|))
                                                                          |recordLst|)
                                                                  (_@ |:record| |record| |recordLst|)
                                                          |else|   (PUTASSOC |destination|
                                                                          (APPEND (CDR (ASSOC
|destination|
|recordLst|))
                                                                                  (LIST (LIST |ref|

|finishTime|
|productType|

|orderBatchSize|
                                                                                        )))
                                                                          |recordLst|)
                                                                  (_@ |:record| |recordLst|)))
                        |else|   (  |if|   (AND (NULL (CAR (@ |:record|)))
                                                (NULL (ASSOC |destination| |recordLst|)))
                                      |then|   (PUTASSOC |destination|
                                                      (LIST (LIST |ref| |finishTime|
|productType|
                                                                  |orderBatchSize|))
                                                      |recordLst|)
                                              (_@ |:record| (CDR |recordLst|))
                                      |elseif|   (NULL (ASSOC |destination| |recordLst|))
                                      |then|   (PUTASSOC |destination|
                                                      (LIST (LIST |ref| |finishTime|
|productType|
                                                                  |orderBatchSize|))
                                                      |recordLst|)
                                              (_@ |:record| |recordLst|)
                                      |else|   (PUTASSOC |destination|
                                                      (APPEND (CDR (ASSOC |destination|
|recordLst|))
                                                              (LIST (LIST |ref| |finishTime|
                                                                          |productType|
                                                                          |orderBatchSize|)))
                                                      |recordLst|)
                                              (_@ |:record| |recordLst|)))))))

(|Method| (  (|Collecter| |RestData|)   |self| |TypeLst| |typeAndBatch|) "deal with rest of data"
   (LET ((|type| (CAR |typeAndBatch|))
         (|batchSize| (CADR |typeAndBatch|))
         |newList|)
```

```
(  |if|   |batchSize|
      |then|   (SETQ |newList| (LIST (CDR (ASSOC |type| |TypeLst|))
                                      |batchSize|))
    |else|   (SETQ |newList| (LIST (CDR (ASSOC |type| |TypeLst|)))))
|newList|))

(|Method| (  (|Collecter| |SPIndication|)   |self|)
   "fetch WIP data been colected and then indicate the system state by analysis the data."
   (LET ((|data| (CAR (@ |:record|)))
         (|intervals| (_ (@ |:drawing|)
                         |PromptForWord| "How many intervals do you interested? "))
         |WIPData| |WIPHistData|)
        (SETQ |WIPData| (_ |self| |GetWIPData| |data|))
        (SETQ |WIPHistData| (_ |self| |SState?| |WIPData| |intervals|))
        (_ |self| |DSPWIPHistT| |WIPHistData| |WIPData|)))

(|Method| (  (|Collecter| |SSIndication|)   |self|)
   "fetch WIP data been colected and then indicate the system state by analysis the data."
   (LET ((|data| (CAR (@ |:record|)))
         (|intervals| 2)
         |WIPData| |WIPHistData|)
        (SETQ |WIPData| (_ |self| |GetWIPData| |data|))
        (SETQ |WIPHistData| (_ |self| |SState?| |WIPData| |intervals|))
        (_ |self| |DSPWIPHistT| |WIPHistData| |WIPData|)))

(|Method| (  (|Collecter| |SState?|)   |self| |data| |intervals|)
   "analyze histogram of WIP, thus user can determine whether the system has reached steady
state"
   (PROG* ((|size| (LENGTH |data|))
           (|medium| (FIX (QUOTIENT |size| |intervals|)))
           (|startP| 1)
           (|endP| (IDIFFERENCE (IPLUS |medium| |startP|)
                                1))
           (|intervalP| 1)
           |WIPList| |WIPSubList|)
          (  |while|   (NEQ |intervalP| |intervals|)
             |do|   (SETQ |WIPSubList| (LIST NIL))
                 (  |for|   \i   |from|   |startP|   |to|   |endP|
                    |do|   (LET ((|key| (CAAR (NTH |data| \i))))
                                (  |if|   (NEQ (ASSOC |key| |WIPSubList|)
                                               NIL)
                                      |then|   (PUTASSOC |key| (IPLUS (CDR (ASSOC |key|

|WIPSubList|))
                                                                                 1)
                                                                |WIPSubList|)
                                      |else|   (PUTASSOC |key| 1 |WIPSubList|))))
                 (SETQ |startP| (IPLUS |startP| |medium|))
                 (SETQ |endP| (IDIFFERENCE (IPLUS |medium| |startP|)
                                           1))
                 (SETQ |intervalP| (ADD1 |intervalP|))
                 (  |if|   |WIPList|
                      |then|   (SETQ |WIPList| (APPEND |WIPList| (LIST (CDR |WIPSubList|))))
                    |else|   (SETQ |WIPList| (LIST (CDR |WIPSubList|)))))
          (SETQ |WIPSubList| (LIST NIL))
          (  |for|   \i   |from|   |startP|   |to|   |size|
             |do|   (LET ((|key| (CAAR (NTH |data| \i))))
                         (  |if|   (NEQ (ASSOC |key| |WIPSubList|)
                                        NIL)
                               |then|   (PUTASSOC |key| (IPLUS (CDR (ASSOC |key| |WIPSubList|))
                                                                1)
                                                        |WIPSubList|)
                               |else|   (PUTASSOC |key| 1 |WIPSubList|))))
          (  |if|   |WIPList|
               |then|   (SETQ |WIPList| (APPEND |WIPList| (LIST (CDR |WIPSubList|))))
             |else|   (SETQ |WIPList| (LIST (CDR |WIPSubList|))))
          (RETURN |WIPList|)))

(|Method| (  (|Collecter| |SaveData|)   |self|) "save data into a file which is with extend."
   (LET ((|fileStream| (OPENSTREAM (_ |self| |GetMetaIndexFileName|)
                                   'OUTPUT
                                   'NEW))
         (|description| (_ |self| |GetMetaDescription|)))

  (* |;;;| "print the description string")

       (PRIN2 |description| |fileStream|)

  (* |;;;| "write out the data ")
```

```
        (PRIN2 (@ |:log|)
               |fileStream|)

  (* |;;;| "close the file")

        (CLOSEF |fileStream|)))

(|Method| (  (|Collecter| |SaveDataIfWanted|)   |self|)
   "Save off the data if the user confirms that he wants to save."
   (_ (@ |:drawing|)
      |ClearPromptWindow|)
   (  |if|   (MOUSECONFIRM "About to save the flow data to disc" NIL (_ |self| |\\GetBestMsgStream|))
         |then|                                                    (* \; "save if off")
             (_ |self| |SaveData|)
       |else|                                                      (* \;
                                                                   "get rid of the data already stored")
           (_ |self| |ExpungeData|)))

(|Method| (  (|Collecter| |Start|)   |self|)
   "check initial setting, if T - start, otherwise tell user to select them."
   (LET ((|IOPath| (@ |:flowList|))
         (|processNodes| (@ |:attachees|))
         (|boxes| (@ |:boxList|)))
        (_@ |:alreadyReset?| NIL)
        (_ (@ |:drawing|)
           |ClearPromptWindow|)
        (  |if|   (AND |IOPath| (OR |processNodes| |boxes|))
             |then|                                                (* \;
                                        "prompt menu for select independent variable(s), and then
start")
                  (_ (@ |:drawing|)
                     |PrintMessage|
                     "To generate metamodel, choose independent variable(s) from the menu.")
                  (_ |self| |InitialLog|)
                  (|_Super| |self| |Start|)
           |elseif|   |IOPath|
             |then|                                                (* \;
                                       "mention user to set input and output paths before running the
model")
                  (_ (@ |:drawing|)
                     |PrintMessage| "Choose process nodes before running the model.")
           |elseif|   (OR |processNodes| |boxes|)
             |then|                                                (* \;
                                        "mention user to choose process nodes before running the
model")
                  (_ (@ |:drawing|)
                     |PrintMessage| "Choose input and output paths before running the model.")
           |else|   (_ (@ |:drawing|)
                       |PrintMessage|
                       "Choose process nodes, and input and output paths for this run."))))

(|Method| (  (|Collecter| |StoreValue|)   |self|) "save data into a file."
   (  |if|   (@ |:record|)
         |then|   (LET ((|stream| (OPENSTREAM (CDAR (@ |:log|))
                                             'OUTPUT
                                             'NEW))
                        (|key| (CAAR (@ |:log|)))
                        (|records| (@ |:record|))
                        (|pathes| (  |for|   \i   |from|   1   |to|   (LENGTH (@ |:flowList|))
                                     |collect|   (@ (CAR (NTH (@ |:flowList|)
                                                              \i))
                                                 |name|)))
                        |inputFlowRecord| |outputFlowRecord|)

  (* |;;;| "dealing with original data to meet user requirment before save on file.")

                  (  |if|   (EQ (LENGTH (@ |:flowList|))
                                2)
                       |then|                                      (* \;
                                                                   "only one input path has been chosen")
                           (SETQ |inputFlowRecord| (CAR |records|))
                           (SETQ |outputFlowRecord| (CADR |records|))
                           (SETQ |newFlowRecord| (_ |self| |ProcessData| |inputFlowRecord|
NIL
                                                                    NIL |outputFlowRecord| |key|))
                     |elseif|   (EQ (LENGTH (@ |:flowList|))
                                    3)
                       |then|                                      (* \;
                                                                  "there are two input paths been
selected")
```

```
        (SETQ |input1FlowRecord| (CAR |records|))
        (SETQ |input2FlowRecord| (CADR |records|))
        (SETQ |outputFlowRecord| (CADDR |records|))
        (SETQ |newFlowRecord| (_ |self| |ProcessData| |input1FlowRecord|
                                        |input2FlowRecord| NIL
                                        |outputFlowRecord| |key|))
|elseif|   (EQ (LENGTH (@ |:flowList|))
              4)
  |then|                                (* \;
                                          "there are three input paths been
selected")
                                 (SETQ |input1FlowRecord| (CAR |records|))
                                 (SETQ |input2FlowRecord| (CADR |records|))
                                 (SETQ |input3FlowRecord| (CADDR |records|))
                                 (SETQ |outputFlowRecord| (CADDDR |records|))
                                 (SETQ |newFlowRecord| (_ |self| |ProcessData| |input1FlowRecord|
                                                          |input2FlowRecord|
|input3FlowRecord|
                                                          |outputFlowRecord| |key|)))
                                                        (* \; "write data on file")
                        (PRIN2 |newFlowRecord| |stream|)

  (* |;;;| "can be used straight away")

                        (|OperateOnLogFiles| (@ |:log|)
                              '|CloseLogFile| T))))

(|Method| (  (|Collecter| |SwitchOff|)   |self|) "switch off self."
   (|_Super| ))

(|Method| (  (|Collecter| |SwitchOn|)   |self|)
   "switch self on at first, then attach self to all the nodes of the map"
   (LET* ((|page?| (@ |:drawing|))
          (|objLst| (@ |page?| |objectList|))
          (|exits| (  |for|   |joints|   |in|   |objLst|   |collect|   |joints|
                      |when|   (_ |joints| |InstOf!| 'IDEF0TAG))))
        (_ (@ |:drawing|)
           |ClearPromptWindow|)
        (  |if|   (AND |page?| (_ |page?| |InstOf!| 'IDEMTRANSITIONMAP))
           |then|   (  |if|   (EQ (LENGTH |exits|)
                                  2)
                       |then|

  (* |;;;| "only one thred on the zoomin page so that the collecter attach all zooming nodes of the
page automatically")

                                 (_ |self| |AttachZoomNode|)
                                 (_ |self| |ChoosePairOfPath|)
                                 (_ (@ |:drawing|)
                                    |PrintMessage|
                        "All nodes on this page that you wish to collect data from has been
attached.")
                           |elseif|   (MOUSECONFIRM "Do you want to attach all nodes on the page?"
                                              NIL (_ |self| |\\GetBestMsgStream|))
                       |then|

  (* |;;;| "more than one thred on the zooming page, but all nodes are still of interest so that
collecter attachs itself on")

                                 (_ |self| |AttachZoomNode|)
                                 (_ (@ |:drawing|)
                                    |PrintMessage|
                                 "Now attach all input pathes first, and then attach output
pathes.")
                                 (  |for|   \i   |from|   1   |to|   (LENGTH |exits|)
                                      |do|   (_ |self| |ChooseAnObj|))
                                 (_ (@ |:drawing|)
                                    |PrintMessage|
                        "All nodes on this page that you wish to collect data from has been
attached.")
                           |else|

  (* |;;;| "more than one thred on the zoomin page, so the collecter only attach those zooming
nodes selected by user.")

                                (_ |self| |AttachPartialZoomNode|)
                                (_ (@ |:drawing|)
                                   |PrintMessage|
                                   "The nodes you selected on this page has been attached."))
           |elseif|   (AND |page?| (_ |page?| |InstOf!| 'IDEF0NEWPAGE))
```

```
                |then|    (_ |self| |AttachIdemNode|)
                     (_ (@ T:drawing|)
                        |PrintMessage| "All boxes on this page that you wish to collect data from
has been attached. Before initialising simulation, choose input and output path by object mene."
)
               |else|    (ERROR |self| " has been located on wrong page."))))
   (|_Super| )
   (_ |self| |EventChanged|))

(|Method| (  (|Collecter| |TakeRecord|)   |self| |event|) "record data that user request on self."
   (LET* ((|controller| (@ |controller|))
          (|time| (@ |controller| |timeNow|))
          (|ref| (@ (@ (@ |event| |package|)
                       |datum|)
                    |reference|))
          (|destination| (@ (@ (@ |event| |package|)
                               |datum|)
                            |destination|))
          (|pathNames| NIL)
          (|recordLst| (@ |:record|))
          (|whatToRecord| (@ |:IndVar|))
          |pathName|)
         (  |for|    \i    |from|    1    |to|    (LENGTH (@ |:flowList|))
             |do|    (SETQ |pathName| (@ (CAR (NTH (@ |:flowList|)
                                                   \i))
                                          |name|))
                  (  |if|    |pathName|
                        |then|    (SETQ |pathNames| (APPEND |pathNames| (LIST |pathName|)))
                     |else|    (SETQ |pathNames| (LIST |pathName|))))
         (  |if|    (NOT (EQUAL |destination| (CAR (LAST |pathNames|))))
             |then|    (  |if|    (AND (NULL (CAR (@ |:record|)))
                                       (NULL (ASSOC |destination| |recordLst|)))
                            |then|    (PUTASSOC |destination| (LIST (LIST |ref| (@ |:numberOfQ|)
                                                                        |time|))
                                                |recordLst|)
                                      (_@ |:record| (CDR |recordLst|))
                         |elseif|    (NULL (ASSOC |destination| |recordLst|))
                           |then|    (PUTASSOC |destination| (LIST (LIST |ref| (@ |:numberOfQ|)
                                                                        |time|))
                                                |recordLst|)
                                      (_@ |:record| |recordLst|)
                         |else|    (PUTASSOC |destination| (APPEND (CDR (ASSOC |destination|
                                                                                |recordLst|))
                                                                     (LIST (LIST |ref| (@
|:numberOfQ|)
                                                                                |time|)))
                                                |recordLst|)
                                    (_@ |:record| |recordLst|))
                 |else|                                          (* \; "to record finish time")
                     (LET ((|finishTime| (@ |event| |qEntryTime|))
                           (|productType| (CAR (@ (@ (@ |event| |package|)
                                                     |datum|)
                                                  |data|)))
                           (|orderBatchSize| (CADR (@ (@ (@ |event| |package|)
                                                         |datum|)
                                                      |data|))))
                          (  |if|    (EQ |whatToRecord| 1)
                              |then|    (  |if|    (AND (NULL (CAR (@ |:record|)))
                                                        (NULL (ASSOC |destination| |recordLst|)))
                                             |then|    (PUTASSOC |destination| (LIST (LIST |ref|
|finishTime|)
                                                                                         )
                                                                |recordLst|)
                                                        (_@ |:record| (CDR |recordLst|))
                                           |elseif|    (NULL (ASSOC |destination| |recordLst|))
                                             |then|    (PUTASSOC |destination| (LIST (LIST |ref|
|finishTime|)
                                                                                         )
                                                                |recordLst|)
                                                        (_@ |:record| |recordLst|)
                                           |else|    (PUTASSOC |destination|
                                                                (APPEND (CDR (ASSOC |destination|
                                                                                    |recordLst|))
                                                                        (LIST (LIST |ref| |finishTime|)))
                                                                |recordLst|)
                                                      (_@ |:record| |recordLst|))
                                                                        (* \; "to record finish time only")
                                   |elseif|    (EQ |whatToRecord| 2)
```

```
                                |then|                                    (* \;
                                                                          "to record finish time and product
type")
                                    (  |if|   (AND (NULL (CAR (@ |:record|)))
                                                   (NULL (ASSOC |destination| |recordLst|)))
                                         |then|   (PUTASSOC |destination| (LIST (LIST |ref| |finishTime|
|productType|))
                                                                |recordLst|)
                                                 (_@ |:record| (CDR |recordLst|))
                                       |elseif|   (NULL (ASSOC |destination| |recordLst|))
                                         |then|   (PUTASSOC |destination| (LIST (LIST |ref| |finishTime|
|productType|))
                                                                |recordLst|)
                                                 (_@ |:record| |recordLst|)
                                       |else|   (PUTASSOC |destination|
                                                          (APPEND (CDR (ASSOC |destination|
|recordLst|))
                                                                  (LIST (LIST |ref| |finishTime|
                                                                                |productType|)))
                                                          |recordLst|)
                                                 (_@ |:record| |recordLst|))
                                |elseif|   (EQ |whatToRecord| 3)
                                  |then|                                  (* \;
                                                                          "to record finish time and order batch
size")
                                    (  |if|   (AND (NULL (CAR (@ |:record|)))
                                                   (NULL (ASSOC |destination| |recordLst|)))
                                         |then|   (PUTASSOC |destination|
                                                                (LIST (LIST |ref| |finishTime|
|productType|
                                                                                |orderBatchSize|))
                                                                |recordLst|)
                                                 (_@ |:record| (CDR |recordLst|))
                                       |elseif|   (NULL (ASSOC |destination| |recordLst|))
                                         |then|   (PUTASSOC |destination|
                                                                (LIST (LIST |ref| |finishTime|
|productType|
                                                                                |orderBatchSize|))
                                                                |recordLst|)
                                                 (_@ |:record| |record| |recordLst|)
                                       |else|   (PUTASSOC |destination|
                                                          (APPEND (CDR (ASSOC |destination|
|recordLst|))
                                                                  (LIST (LIST |ref| |finishTime|
                                                                                |productType|
|orderBatchSize|
                                                                                )))
                                                          |recordLst|)
                                                 (_@ |:record| |recordLst|)))))))

(|Method| (  (|Collecter| |\\GetFileStream|)   |self| |keys|) "Method documentation."
   (LET ((DIRECTORIES NIL)
         (|logEntry| (|GetMultiAssoc| |keys| (@ |:log|))))
        (  |if|   (STREAMP |logEntry|)
```

```
            |then|    |logEntry|
          |elseif|    (NULL |logEntry|)
            |then|    (SETQ |logEntry| (OPENSTREAM (_ |self| |\\GetLogFileName| |keys|)
                                                   'OUTPUT
                                                   'NEW))
                  (_@ |:log| (|PutMultiAssoc| |keys| |logEntry| (@ |:log|)))
                  |logEntry|
          |else|    (ERROR |logEntry| "is not a valid file stream for logging information"))))

(|Method| (  (|Collecter| |\\GetLogFileName|)   |self| |keys|) "Method documentation."
   (LET ((|name| (APPLY 'PACK*
                        (CONS (@ |:controller:thisID|)
                              (CONS '- (NCONC1 (  |bind|   |label|   |for|   |key|   |in|   |keys|
                                                  |when|   (SETQ |label|
                                                            (  |if|   (ATOM |key|)
                                                                  |then|   |key|
                                                                |else|   (AND (  |type?|
                                                                                 |instance|
                                                                                 |key|)
                                                                        (|_Try|
                                                                         |key|
|GetName|))))
                                                  |join|   (LIST |label| '-))
                                            (ROOTDATE))))))
        (PACKFILENAME 'NAME (REPLACE-CHARS |name| '((\ . -)
                                                    (\ . -)
                                                    ({ . -)
                                                    (} . -)
                                                    (< . -)
                                                    (> . -)
                                                    (\; . -)))
              'EXTENSION
              'IDEM-META-DATA)))

(|\\UnbatchMethodDefs|)
(DEFINEQ

)
(PUTPROPS COLLECTER.SRC COPYRIGHT ("B.YU" 1994 1995))
(DECLARE\: DONTCOPY
  (FILEMAP (NIL (99155 99166))))
STOP
```