A KNOWLEDGE–BASED APPROACH TO THE DESIGN, SIMULATION, AND EVALUATION OF FLEXIBLE MANUFACTURING SYSTEMS

by

THIRUVENGADAM RAVI, B.E.(Honors), M.A.Sc.

A Thesis

Submitted to the School of Graduate Studies

In Partial Fulfillment of the Requirements

For the Degree

Doctor of Philosophy

McMaster University

April 1994

A KNOWLEDGE–BASED APPROACH TO THE DESIGN, SIMULATION, AND EVALUATION OF FLEXIBLE MANUFACTURING SYSTEMS

TO MY WIFE PRASANNA

DOCTOR OF PHILOSOPHY (1994) McMASTER UNIVERSITY
(Mechanical Engineering) Hamilton, Ontario

TITLE: A Knowledge–Based Approach to the Design, Simulation, and Evaluation of Flexible Manufacturing Systems

AUTHOR: Thiruvengadam Ravi
B.E.(Honors) (University of Madras, India)
M.A.Sc. (University of Windsor, Canada)

SUPERVISOR: Dr. Hoda A. ElMaraghy
Professor and Director
Centre for Flexible Manufacturing Research and Development

NUMBER OF PAGES: xvii, 271

ABSTRACT

Launching new manufacturing systems for production is a difficult task which involves several people and resources, and is time–consuming and capital intensive. Flexible manufacturing systems (FMS), which are a new generation of manufacturing systems, further complicate this launching process because they consist of several interacting components such as workstations, pallets, automated material handling systems, and buffers. The lack of knowledge about the interaction between these components has often resulted in the poor performance of many FMS designs. Major corporations today are adopting a proactive philosophy rather than a reactive one, and for a complex and capital intensive manufacturing system such as an FMS, the design process is extremely critical for its successful performance.

The design process, which precedes the launching of an FMS, is iterative and consists of planning, model development, and output analysis. While several computer–based tools are available for modeling and evaluating FMS designs, there is a dearth of such tools for synthesizing new designs and analyzing output from simulation models to improve these designs. System designers have traditionally performed these two activities using their knowledge and experience. These are cumbersome and time–consuming activities and consequently increase the design cycle time.

The objective of this thesis is to reduce the FMS design cycle time by automating the design process. A knowledge–based system called FMX, Flexible Manufacturing Expert, was developed in this thesis to generate and evaluate designs of manufacturing systems with a high degree of automation. The object–oriented FMX is an intelligent system which combines expert systems and simulation

modeling to design and evaluate flexible manufacturing systems. It consists of several modules, the major ones including an expert design synthesizer to generate initial system designs, a simulation model developer to automatically convert designs into graphical simulation models, and an expert analyzer to analyze simulation output, identify design deficiencies, and recommend changes. The current implementation of FMX was developed using the Knowledge Engineering Environment (KEE) expert system shell and the SimKit simulation package, and it runs on a Sun workstation under a UNIX operating system.

FMX has been applied for designing flexible manufacturing systems in the machining domain. Two case studies have been included in this thesis to demonstrate its capabilities—one of which includes industrial system used by a major automotive manufacturer in Ontario. The first case study took two iterations while the second one took three iterations to fine tune a synthesized design that satisfies various performance measures such as production volumes, equipment utilizations, equipment blocking, and queue length of automatic pallet changers. For both case studies, the investment cost per part index decreased for each iteration thus proving that the design changes suggested by the expert analyzer improved the performance of the initial system design while reducing its cost. The final output from FMX is a list of components that comprise the system, its graphical layout, and its performance measures.

FMX is a comprehensive decision support system which integrates all phases of a flexible manufacturing system design into a single software framework. Its modules such as the expert design synthesizer, simulation model developer, and expert analyzer each address a specific phase of the system design process. FMX generates consistent designs with minimum intervention from the user and hence reduces the design cycle time significantly.

ACKNOWLEDGEMENTS

I'm very grateful to my supervisor, Dr. Hoda A. ElMaraghy for her valuable guidance and support throughout this research. The research scholarships awarded by Dr. Hoda A. ElMaraghy and the financial support from the Department of Mechanical Engineering in the form teaching assistantships are greatly appreciated.

Special thanks to all my friends and fellow graduate students for their constant advice and encouragement. I'm also very thankful to all members of the Centre for Flexible Manufacturing Research and Development for their help in creating a stimulating and friendly work environment.

Finally, I'd like to thank all my family members for their continued support, especially my wife Prasanna, whose patience, encouragement, and moral support has made it all possible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Flexible manufacturing systems (FMS) are automated manufacturing systems which have the flexibility of job shops and the efficiency of mass production systems. These are integrated systems which offer significant benefits in the mid–volume and mid–variety type of batch manufacturing environment. They are complex consisting of several interacting components, the major ones being workstations to process the parts, automated material handling system to move the parts between workstations, pallets to carry the parts, and buffers to store the parts at each workstation. The lack of knowledge about the performance of these components as an integrated system has often led to dismal performance of many FMS designs. In addition, since flexible manufacturing systems involve a huge amount of capital, the FMS design process is very crucial for the successful performance of any flexible manufacturing system after its installation.

The design of a flexible manufacturing system is an iterative process and consists of the following activities: planning, design synthesis, model development, and output analysis. Several modeling tools are available to model and evaluate the performance of FMS designs. Discrete simulation models, in particular, have gained a wide acceptance over analytical and physical models because of their ability to handle realistic information and be flexible. Several computer–based FMS

simulators are available for modeling and evaluating the performance of an FMS design. There is, however, a dearth of computer–based tools for synthesizing new FMS designs and analyzing outputs from models to improve these designs. Traditionally, system designers have been responsible for performing these two activities using their knowledge and experience. Hence, these activities are cumbersome and time–consuming, and consequently increase the design cycle time.

The objective of this thesis is to automate and integrate the overall FMS design process, thus: (i) reducing the design cycle time, and (ii) generating consistent and better FMS designs. A knowledge–based system called FMX, Flexible Manufacturing Expert, has been developed in this thesis to achieve this objective.

The object–oriented FMX combines expert system techniques and discrete simulation methodology for designing and evaluating flexible manufacturing systems. FMX consists of the following modules:

(i) the object–oriented ***component library*** which contains a hierarchical description of flexible manufacturing system components,

(ii) the ***design synthesizer*** which contains the heuristic and analytical knowledge in the form of rules and methods to reason about the various objects and generate flexible manufacturing system designs,

(iii) the ***simulation model developer*** which converts the design generated by the design synthesizer to a graphical simulation model,

(iv) the ***simulator*** which simulates the model and generates output,

(v) the ***analyzer*** which analyzes the output from the simulator to identify design deficiencies and suggests suitable changes to meet the specified performance measures, and

(vi) the ***user interface*** which enables communication with the various modules.

The input to FMX is information about the family of part types and the system requirements. The final output from FMX is a flexible manufacturing system design for processing the part family.

This thesis is organized as follows: Chapter 2, entitled **Literature Review,** provides an overview of FMS and their components. It explains the various stages of the FMS design process and reviews previous research efforts in this area. It also discusses the drawbacks of the current approach and presents the motivation and objectives of this research.

Chapter 3, entitled **Expert Systems and Simulation,** includes the various approaches of combining expert systems and discrete–event simulation. It also briefly discusses some concepts in object–oriented programming and knowledge–based simulation systems. Finally, in addition to an overview of the FMS system structure, it also describes the object–oriented modeling of the system components in detail.

Chapter 4, entitled **Design Synthesizer,** explains the different types of input information required to drive the FMS design synthesis process. It describes this process in detail and presents the heuristic and analytical knowledge required for generating an initial FMS design. Finally, it elaborates upon the implementation of the design synthesizer module of the FMX prototype using the Knowledge Engineering Environment (KEE) software.

Chapter 5, entitled **Model Developer and Simulator,** explains the two major activities of discrete–event simulation: model development and model execution. It describes the various principles involved in automatically developing a simulation model of an FMS design. It presents the implementation of the simulation model developer module of the FMX prototype using the KEE and SimKit software.

Finally, this chapter describes the model execution activity and presents the implementation of the simulator module of FMX.

Chapter 6, entitled **Analyzer,** begins with a description of the analysis of simulation output. It then outlines the functions of the expert analyzer developed in this thesis and describes the various performance measures which evaluate the goodness of a flexible manufacturing system design. It also discusses the strategy used to identify deficiencies in the FMS designs and the changes recommended to overcome these deficiencies. Finally, it presents the implementation of the analyzer module of the FMX prototype.

Chapter 7, entitled **Case Studies and Discussion,** illustrates the usefulness of the FMX system with the help of two case studies. One of these case studies is an industrial example which involves the machining of brake drums. This chapter presents initial FMS designs synthesized by the design synthesizer module of FMX and discusses the responses of the design synthesizer to variations in input data for each case study. It also discusses the changes made to the original design of each case study by other modules of FMX for each design iteration. It finally shows the design which satisfies the performance measures of each case study.

Chapter 8, entitled **Conclusions,** presents the conclusions of this research effort and discusses its contributions. It presents the various features of FMX which makes it a comprehensive decision support tool. It then summarizes the achievements of this work and compares FMX with other similar systems. Finally, it discusses the limitations of this work and highlights the areas which should be addressed in the future to overcome these limitations.

CHAPTER 2

LITERATURE REVIEW

This chapter begins with an overview of flexible manufacturing systems and proceeds to describe the various components of these systems. It also describes in detail the various stages of the flexible manufacturing system design process. Previous research efforts in this area are reviewed and their drawbacks are discussed. Finally, the objectives of this research are presented and the various research issues are identified.

2.1 Flexible manufacturing systems

Until the last decade, the two major types of manufacturing systems were job shops and mass production systems. While job shops process a high variety of parts at low volumes, mass production systems, in contrast, process a low variety of parts at high volumes. In the last decade, however, there has been an increased demand for products with higher quality, more variety, and lower costs. This has forced industries to evaluate their existing manufacturing systems and develop new ones which would be both flexible and adaptive—systems that would possess the flexibility of job shops and the efficiency of mass production systems. The result of such a need was the development of the flexible manufacturing system.

Flexible manufacturing systems, a subset of Computer Integrated

Manufacturing (CIM) systems, are considered to be the building blocks for the 'factory of the future' or 'the unmanned factory'. While CIM refers to the integration of all functions from marketing to manufacturing to shipping using computers, flexible manufacturing deals only with some functions such as manufacturing, material handling, and tooling. A flexible manufacturing system is described as "an implementation of a Computer Integrated Manufacturing (CIM) system at the shop floor level [1]."

The flexible manufacturing methodology was first applied to the machining environment. However, years later, it has been extended to other manufacturing processes such as assembly, casting, molding, sheet metal forming, and welding [2]. Currently, the major users of this methodology include aerospace, defense, automotive, and light and heavy engineering industries. Many flexible manufacturing systems are already operational in industrialized countries such as Canada, Japan, U.K, U.S.A, and Germany. An increase in revenues, from $769 million in 1990 to $1.7 billion in 1997, has been predicted for these systems [3].

Flexible manufacturing systems are integrated systems of workstations and automated material handling systems (MHS) under computer control for the automatic random processing of palletized parts [4]. They are automated and are capable of processing a family of part types simultaneously. Major benefits offered by FMS are:

(i) reduced direct labor costs,

(ii) high product quality,

(iii) adaptation to changing production environments, and

(iv) the ability to maintain production despite equipment failures.

2.2 Components of an FMS

A flexible manufacturing system is a complex manufacturing system consisting of many interacting components—the major ones being workstations, automated material handling systems, pallets, and work–in–process storage systems (also known as buffers). The major FMS components are shown in Figure 2.1.

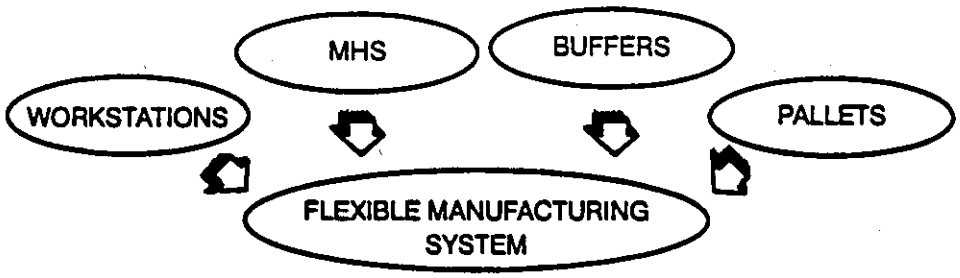


Figure 2.1. Generic FMS components

Workstations which form the nucleus around which the system is built are further classified into primary and secondary workstations. Primary workstations, such as numerically controlled machine tools, perform various metal cutting processes such as turning, drilling, reaming, boring, and milling. Secondary workstations, on the other hand, perform support activities such as loading and unloading parts to and from the pallets, inspecting the parts, and washing the parts.

Automated material handling systems transport the pallets from one workstation to another in the system. The types of material handling systems commonly used in FMS are conveyors, automated guided vehicles (AGVs), and rail carts.

Work–in–process storage systems act as a buffer between the material

handling system and the workstation. Their purpose is to keep the pallet waiting for a workstation or material handling system instead of a workstation or material handling system waiting for the pallet. Such a strategy effectively increases the utilization of the equipment. Automatic pallet changers are the most commonly used storage systems and each workstation is provided with an automatic pallet changer of a certain storage capacity.

Before parts are sent into the system for processing, they are fixed on pallets. A pallet is a steel disc or a square with surface slots which are used to fasten the fixture to the pallet. Fixtures provide a permanent orientation of the part and can be built for a single part type or a family of part types. They are attached to the pallet permanently and hence, additional or new fixtures always require new pallets.

The size of a flexible manufacturing system is determined by the number of workstations in the system for processing parts. In a survey of 53 systems, it was found that the smallest systems had 2 to 3 workstations while the largest ones had 13 workstations or more [5]. Although much is known about the properties and behavior of the various components in an FMS, very little is known about the operation of these individual components as an integrated system. This lack of knowledge about the dynamics of the system has increased the complexity of the FMS design procedure.

2.3 FMS design procedure

The design of any product, process, or system comprises of the following three phases as shown in Figure 2.2: formulation, synthesis, and evaluation [6]. In the design formulation phase, the requirements and specifications of the design

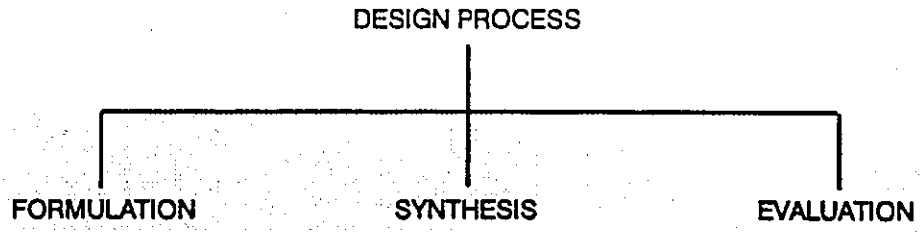


Figure 2.2. The three phases of a design process

problem are established, and the result of this phase is the definition of a design problem. Design synthesis identifies one or more design solutions which are consistent with the design formulation. Design evaluation analyzes a completely specified design to check its performance. Typically, a designer formulates the design problem, synthesizes one or more solutions, and then evaluates these solutions. The design process, however, is not sequential; often, it requires iterations between these various phases. The flexible manufacturing design process is very similar and consists of the following phases:

(i) planning,

(ii) design synthesis,

(iii) model development, and

(iv) analysis.

In the planning (design formulation) phase, the family of part types for which a new FMS is to be designed is determined. The process plan for each part type in the family is developed and the corresponding production volumes are determined. An initial FMS design is proposed during the design synthesis phase. A model of this design is then developed and its output analyzed to see if the design meets the objectives. If it does, then the design is implemented. Otherwise, the

problems in the design are identified and appropriate changes are made to it. A model is developed again for this new design and another set of output is generated. The modeling and analysis phases, which form the design evaluation phase, are repeated until the objectives are met. The FMS design process is, therefore, iterative as illustrated by the flow chart in Figure 2.3.

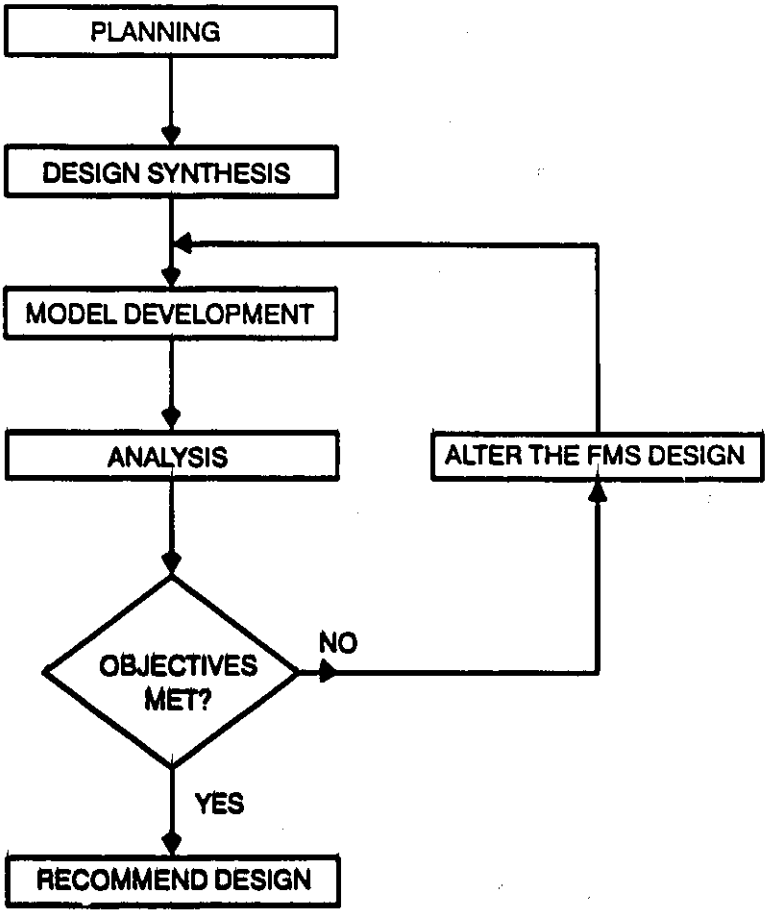


Figure 2.3. Various phases of the FMS design process

2.3.1 Design synthesis

The basic design synthesis problem, given a set of elements and a set of constraints among these elements, seeks to assemble a structure from the elements that satisfy the constraints [7]. The design synthesis of a flexible manufacturing system involves selecting the various components, determining their quantity, and combining them to form the system with a suitable layout. The successful performance of an FMS depends to a large extent on the decisions made during the design synthesis phase. This is because it is the design generated in the synthesis phase that is modified and improved later. Traditionally, FMS design synthesis has been the responsibility of the system designer due to the lack of suitable computer–aided synthesis tools [8]. The system designer uses two types of information to generate an initial FMS design:

(i) information on the family of part types and the system requirements of the FMS, and

(ii) information on the various FMS components.

A block diagram of the design synthesis phase is shown in Figure 2.4.

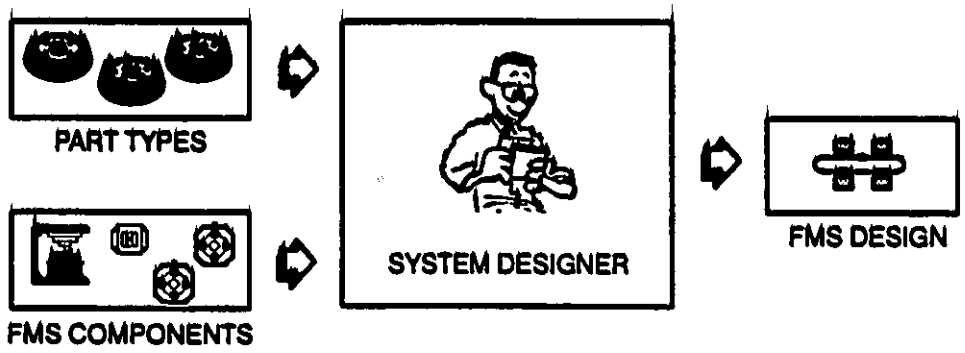


Figure 2.4. Block diagram of the design synthesis phase

In order to generate an initial design, system designers should have a complete understanding of the behavior of flexible manufacturing systems and their components. They process large amounts of information and use their knowledge and experience to make the correct decisions. For example, to select a machine from a feasible set of machines for processing a family of parts, the system designer uses heuristics (or rules) to match the features of the parts with the corresponding features of a machine. Hence, the design synthesis phase is knowledge–intensive and consequently, is a tedious and time–consuming task.

The emergence of expert systems from the laboratory to the real world has led to the increased popularity of artificial intelligence (AI). Expert systems mimic the performance of one or more human experts in a specific domain. They have the potential to model the knowledge and experience of the system designer in a computer. This computer model could then be used to aid the decision–making process. In recent years, the use of expert systems for the synthesis of flexible manufacturing systems has been the focus of researchers.

Parthasarathy and Kim [9] discussed some critical issues in representing and utilizing knowledge for designing intelligent manufacturing systems (IMS). They proposed a formal framework based on the principles of information theory, transformation theory of production processes, automata theory, and computation theory. They identified the two critical knowledge components of an IMS as decision rules and performance metrics. The objective of their approach was to develop a set of generalized rules to synthesize an IMS. These rules would then be validated by simulating a model of the system under various conditions using a set of performance metrics. The result is the development of a knowledge–based advisor, containing both algorithmic and heuristic knowledge, which would help synthesize factories. While this work outlined the various issues about the design

of intelligent manufacturing systems and their importance, it lacked the development and implementation of a comprehensive knowledge–based system. Nevertheless, it addressed the importance and need for a computer–based system for design synthesis.

2.3.2 Model development

Since an FMS requires a large amount of capital investment, it is very expensive to install an FMS and evaluate its performance. Modeling an FMS design is a cost–effective alternative to evaluate the performance of the design prior to the installation of the system. ElMaraghy and Ravi [10] have outlined the various modeling techniques to model and evaluate FMS designs. These are shown in Figure 2.5 and they include:

(i) analytical models,

(ii) physical models, and

(iii) simulation models.

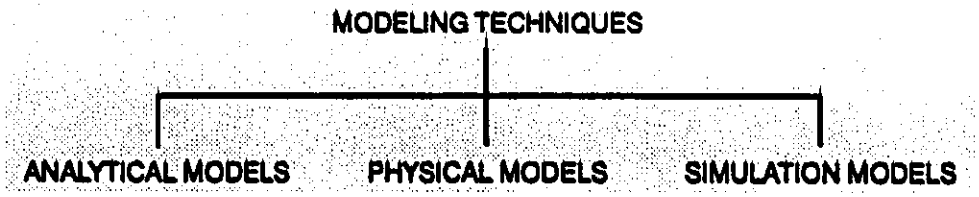


Figure 2.5. Modeling techniques to evaluate FMS designs

2.3.2.1 Analytical models

Analytical models are abstract representations of real world systems in which mathematical variables and expressions represent the physical quantities and

behavior of the actual system. A major portion of analytical models reported in the literature have used queueing network theory [11]. Since these queueing network models provide quick results, they encourage further experimentation with the model. For example, system designers can quickly understand the trade-offs between inventory levels, production requirements, flow times, and equipment reliability for their system by changing design data.

The development of queueing network models, however, is often slow and mathematically intensive because of the extent of the required abstraction from the actual manufacturing system. They are derived using many mathematical assumptions such as exponentially distributed interarrival time for parts, infinite work-in-process storage capacities, first-in-first-out (FIFO) queueing priorities, and exponentially distributed part processing times. Traditionally, the manufacturing community has been skeptical about applying these models to the practical world. However, more and more researchers have shown that these models are robust to changes in the underlying mathematical assumptions [12] and are relatively insensitive to errors in parts and equipment data [13].

MANUPLAN is a commercially available software tool that was developed using queueing network models and is very useful in the early stages of the manufacturing system design process [14]. IBM used MANUPLAN for the initial analysis of a factory of the future design to manufacture printed circuit boards [15]. The number and type of equipment, lot sizes, and storage requirements were decided in this phase. A detailed simulation was then conducted to decide the impact of constraining the buffer sizes of various equipment and to select the appropriate material handling configuration.

Huettner and Steudel [16] used MANUPLAN along with LOTUS 123 spreadsheet and STARCELL simulation package for designing and evaluating

manufacturing systems. The objective of their study was to evaluate the quantity and quality of required inputs and generated outputs, and the time required for using each tool compared to the amount of knowledge gained about the system. Gupta [17] designed a manufacturing cell which considers a new heuristic procedure of alternative routes for parts using MANUPLAN. Results from this analysis confirmed that the proposed heuristic was better than the existing methods.

Analytical models are useful tools to be used in the initial design phase, but they are inappropriate for decisions to be taken in a short time and for studying transient phenomena. They are also inaccurate when there is significant blocking in the system due to limited buffer sizes. A detailed simulation which includes system dynamics is always necessary before finalizing the design.

2.3.2.2 Physical models

Physical models, also called physical simulators, are operational scale models of the actual system ([18], [19], and [20]). They use hardware components whose characteristics are similar to those used in the real world system. The models are generally constructed using plastic construction sets or similar modular components. When controlled by a computer, they physically simulate the operation of the actual system and generate data. This data is then used to evaluate the performance of the actual FMS.

Physical models are often used to test and debug the control software of the actual system, develop various manufacturing strategies, and train operating personnel. In addition to cost being their major drawback, the construction of these simulators is tedious and time-consuming. They are relatively inflexible after construction and any modifications to be made involves rewiring and reconstructing

a part of the model or the entire model itself. Often, they fail to represent some of the complex modules of the real world system.

2.3.2.3 Discrete simulation models

Until a few years back, discrete–event simulation modeling was considered 'a tool of last resort' because it required a considerable amount of human and computer effort, and was time–consuming and expensive [21]. Rapid advances in computer hardware and software, however, have made simulation a powerful tool to model and evaluate FMS.

Each simulation process begins with the development of a simulation model of an FMS design. This model is then executed and the output is collected for analysis. A block diagram of the simulation phase is shown in Figure 2.6. A discrete

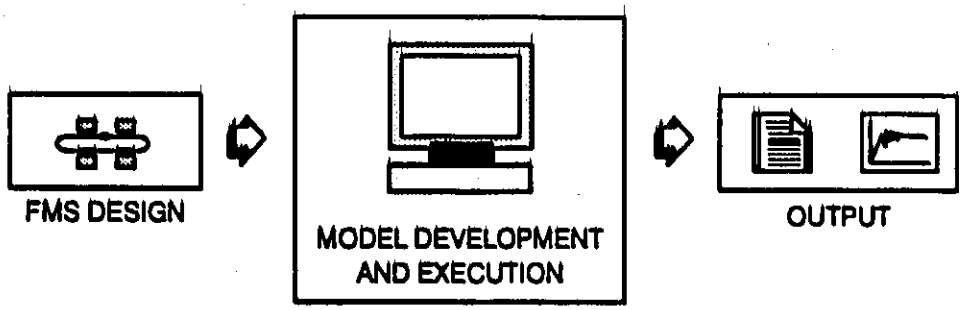


Figure 2.6. Block diagram of the simulation phase

simulation model is a computer program that describes the various operations of a system numerically and logically, rather than mathematically. The program can be easily modified to change system parameters compared to reconstructing and rewiring a physical model. Therefore, simulation models are more flexible and adaptable than physical models. Because simulation models closely duplicate the behavior of the actual system, they provide more realistic information than

analytical models. As a result, discrete simulation models have been used to address a wide variety of problems in the domain of flexible manufacturing systems.

In discrete simulation modeling, the model of the system to be analyzed is built using a programming language. This model then serves as a test bed in which various alternatives are tested and compared. The best alternative is finally selected from this set of alternatives. Two classes of computer languages are available for simulating flexible manufacturing systems [22]:

(i) general–purpose simulation languages, and

(ii) manufacturing simulators.

General–purpose simulation languages, such as GPSS, SLAM, and SIMAN are capable of simulating a variety of systems such as computer systems, traffic systems, communication systems, and manufacturing systems. A model of a system is developed by writing a computer program using the language's modeling constructs such as entities, queues, nodes, resources, and activities. Major drawbacks of these languages are the need for programming expertise, and the long coding and debugging time associated with modeling FMS. Manufacturing simulators, such as MAST, PROMOD, SIMFACTORY, and WITNESS are capable of simulating manufacturing systems only. A major advantage of these simulators is the considerable reduction in model development time because of the use of graphics and menus, and modeling constructs that are specifically related to the components of manufacturing systems. They are, however, limited to modeling only those manufacturing configurations that are allowed by their standard features.

Simulation has been used frequently to address the design and operational problems of flexible manufacturing systems. Several models have been developed using a general–purpose simulation language or a manufacturing simulator [23]. Although they address the problems of specific FMS configurations, they could not

be used to solve the problems of other FMS configurations. Moreover, the modeler has to be familiar with a specific simulation language and its modeling constructs to develop models.

FMS simulators have been developed to overcome this drawback. These simulators can model hardware components and control algorithms of FMS. Several FMS simulators have been developed at universities and research laboratories. These include General Computerized Manufacturing System Simulator—GCMS [24], Flexible Manufacturing Systems Simulator—FMSSIM [25], SIKTAS [26], PATHSIM [27], Graphic Interactive Simulation and Animation—GISA [28], and Modular FMS simulator [29]. They are based on a modular structure and have a good graphics interface. This interface simplifies model building and data input efforts; it also displays output data in the form of bar charts, plots, histograms, and animations. The main advantage of FMS simulators, compared to the previous generation of simulation software, is the reduction in the amount of time and effort required to develop and implement simulation models.

2.3.3 Analysis

Analysis begins with a known structure and reasons about the relationship between its behavior and the elements that make it up [7]. Although simulation is a powerful modeling technique to evaluate the performance of FMS designs, it only provides outputs which describe trends rather than specific solutions. The output from any simulation model consists of numerical data as well as plots, bar charts, histograms, and animations. Traditionally, this output is analyzed by system designers to see if the design meets the performance objectives. If it doesn't, the deficiencies in the design are identified and suitable changes are recommended to overcome these deficiencies. These changes are then implemented in the model and

the simulation process is repeated. The block diagram of the output analysis phase is shown in Figure 2.7.

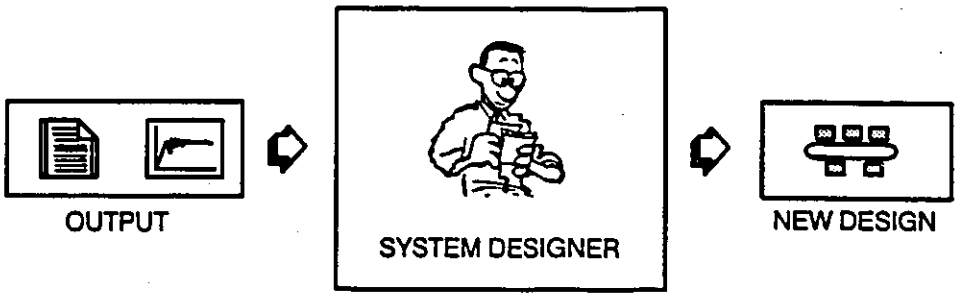


Figure 2.7. Block diagram of the analysis phase

Simulation models lack the capability to analyze output data and provide recommendations that will improve a proposed design. These responsibilities are again borne by the system designer. Consequently, output analysis, like design synthesis, is a time–consuming and knowledge–intensive task. Some researchers have understood the significance of a computer–based system to perform this task and they have developed expert systems for this purpose. These expert systems act as intelligent back ends to conventional simulation models. They analyze the output from the simulation model and recommend improvements to the FMS design if needed.

Lenz and Wichmann [30] developed an expert system called XMAS, in Prolog, to recommend FMS designs based on the analysis of results from the MAST simulation software. A similar prototype expert system called FASDT was developed for the design of flexible assembly systems [31]. FASDT analyzes the output from FASIM, a data–driven simulator for flexible assembly systems, and recommends improvements to the design. Haddock [32] developed a back end in

FORTRAN which interpreted the results of the experimental runs from a SIMAN simulation model and made statistical inferences about various performance measures. However, the expert system which was developed in FORTRAN did not identify design deficiencies in the current design and recommend changes. Moreover, it made the output analysis process cumbersome, especially when the simulated system was large. This was largely due to the structure of SIMAN which presented difficulties while interfacing with the interactive FORTRAN routines.

Mellichamp and Wahab [33] developed a hybrid system that analyzes output from a simulation model, determines whether the operational and financial objectives are met, identifies design drawbacks, and proposes new designs. The expert system was developed using the Knowledge Engineering Environment (KEE) expert system shell and the simulation model was developed in SIMAN. Output from the simulation model is input to the expert system and the design changes recommended by the expert system are then incorporated in the simulation model. The process is repeated until a particular design satisfies both the operational and financial objectives. A major drawback of this approach was the knowledge of SIMAN that the system designer should possess to develop new models or change existing models whenever the expert system recommended changes. Moreover, the interface between the simulation model and expert system was manual—output from the simulation model was manually input to the expert system and changes suggested by the expert system were manually implemented in the SIMAN model. Therefore, this approach was tedious and time-consuming.

Floss and Talavage [34] developed a knowledge-based intelligent manufacturing system design assistant (IMSDA) which analyzes output from model-based analysis tools and provides design recommendations. Two model-based analysis tools were used:

(i) CAN−Q queueing network model [35], and

(ii) GCMS simulator [24].

This resulted in a two−staged approach to the design—the first stage sizes the design using CAN−Q while the second stage performs a detailed design using GCMS. The expert systems for analyzing the output from the CAN−Q model and the GCMS simulator were developed in KEE. A major drawback of IMSDA is the assumption that the quality of the initial design which is input to the CAN−Q model is not critical. The quality of the initial design is critical because it is this design that is modified and improved subsequently. CAN−Q is an analytical model and like other analytical models it is based on several assumptions that do not capture many realistic features of the modeled system. IMSDA failed to exploit the object−oriented capabilities of SimKit, a simulation package that is built on top of KEE, and the common modeling features and graphical capabilities of KEE and SimKit.

Mellichamp et al. [36], developed an expert system called FMS Designer which enhances the work done earlier by Mellicahmp and Wahab [33]. In addition to satisfying the operational and financial objectives, the new system uses a post achievement heuristic to identify improvement opportunities and exploit them. This heuristic approach is employed whenever an existing design does not consume the entire capital investment and it aims to increase the output of one or more part types by making equipment replacements. The computing environment for FMS Designer treats the simulation model and expert system as separate entities. The simulation model was developed in SIMAN on an IBM 3081 mainframe while the expert system was developed in OPS83 [37] on an IBM PC AT machine. Like the previous system, this system expects the system designer to have a good working knowledge of SIMAN. Moreover, the interface between the simulation model and

expert system was manual which made the consultation time–consuming. This is further compounded by the use of a mainframe and a PC based computing environment. While the authors suggested that the interface between the simulation model and the expert system could be automated, they concluded that it was a complex task. This can be attributed to the different programming paradigms used for the simulation model and the expert system and the failure to exploit the similarities between simulation modeling and expert system technology.

Using expert systems as back end for simulation models has excellent potential for analyzing simulation output, identifying design deficiencies, and recommending changes to overcome these deficiencies. To date, very few expert systems have been developed for this purpose. These are, however, not integrated with the simulation model and it is the responsibility of the system designer to manually implement the changes in the simulation model. This increases the overall flexible manufacturing system design cycle time.

2.4 Research objective

The ultimate quality of an FMS design depends on the synthesis of an initial design and on the subsequent refinement of this design by analyzing the simulation output. Traditionally, these activities are performed by systems designers who use their knowledge and experience to recommend good FMS designs. To achieve the level of expertise for recommending FMS designs requires years of experience in designing, modeling, and analyzing flexible manufacturing systems. The types of knowledge required by the system designer are summarized as follows:

(i) Knowledge about the properties and behavior of FMS components, such as workstations, automated material handling systems, buffers, and pallets.

(ii) Knowledge about FMS in general and the interaction between the various system components.

(iii) Knowledge about the various aspects of simulation modeling, such as simulation languages, modeling, computer programming, statistics, and design of experiments.

(iv) Knowledge about analyzing simulation output to identify problems in the design and suggest recommendations to improve the design.

Therefore, flexible manufacturing system design is a knowledge–intensive process and paucity of knowledge in any one of the above–mentioned areas increases the probability of recommending poor FMS designs. Consequently, the design process is time–consuming and increases the FMS design cycle time.

The successful installation of a new FMS depends largely on the design procedure. The design of a new flexible manufacturing system for processing a family of part types is a complex procedure and comprises the following activities: design synthesis, model development, and output analysis. Various modeling techniques, such as analytical, physical, and simulation models, are available to evaluate the performance of proposed FMS designs. Simulation, in particular, has proven to be a powerful tool to model and evaluate FMS designs. Simulation models, however, produce only outputs which describe trends; they do not provide any solutions.

The objective of this research is to combine expert systems and discrete–event simulation to automate the FMS design procedure. Expert system techniques would be used to generate designs, develop simulation models, and analyze output while discrete–event simulation would be used for evaluating the performance of FMS designs. The following steps were identified to achieve this objective:

(i) the development of an expert design synthesizer to generate an initial FMS design,

(ii) the development of a simulation model developer to convert existing FMS designs to graphical simulation models,

(iii) the development of an expert analyzer for analyzing the simulation output, identifying deficiencies in the current design, and recommend changes to the previous design, and

(iv) the integration of the design synthesizer, simulation model developer, simulator, and output analyzer in a single software framework.

Until now, the simulation phase of the FMS design process has been the focus of many researchers ([22], [24], [25], [26], [27], [28], and [29]). However, there are many activities prior to and after simulation that are still done by experienced humans who use their knowledge to aid the decision-making process. This knowledge has always been discarded at the end of a simulation study and reused later when needed.

The use of expert systems for these knowledge-intensive activities is still in its early developmental stages. The research efforts until now have been isolated and rudimentary—for example, the development of a knowledge-based system for design synthesis [9] or the development of a knowledge-based system for output analysis ([30], [31], [32], [33], [34], and [36]). Researchers have overlooked the common features of expert systems and discrete-event simulation in developing a comprehensive decision support system that combine both techniques effectively.

CHAPTER 3

EXPERT SYSTEMS AND SIMULATION

This chapter begins with a brief description of expert systems. It then proceeds to explain in detail the object–oriented and rule–based programming paradigms of expert systems. It also discusses knowledge–based simulation systems which combine expert system techniques and simulation methodology. After providing an overview of the system structure of FMX (Flexible Manufacturing Expert) this chapter concludes with a detailed description of the object–oriented modeling of FMS components.

3.1 Expert systems

An expert system represents the transition from data processing to knowledge processing and is one subset of the work being done in the area of Artificial Intelligence (AI). There has been a tremendous interest in the application of expert system techniques to a wide variety of fields such as finance, engineering, medicine, etc., and this has led to the increased popularity of expert systems in recent years. Freznel [38] defines an expert system as follows:

"An expert system is an advice–giving or a consultation program containing the knowledge and experience of one or more experts in a specific domain that anyone can tap as an aid in solving problems."

The three major components of an expert system, as shown in Figure 3.1, are the knowledge base, inference engine, and user interface. The knowledge base

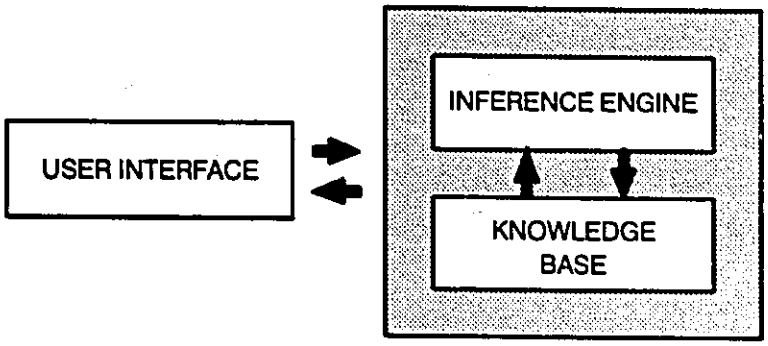


Figure 3.1. Major components of an expert system

contains the expertise or knowledge of an expert or a group of experts in a certain domain. This knowledge is represented in the knowledge base as facts, objects, relationships between objects, rules, and procedures. The inference engine is analogous to the thinking mechanism of the human mind and simulates the problem–solving strategy of a human expert. It uses information provided by the user and information in the knowledge base to solve problems in a certain area. During consultation, the user communicates with the system through the user interface.

Expert systems are essentially computer programs that simulate a human expert's thought process to solve problems. They can be developed in any programming language; some languages, however, are better suited because of their programming paradigm. The various programming paradigms [39] based on which programming languages have been classified are:

(i) procedure–oriented programming paradigm,

(ii) declarative programming paradigm,

(iii) functional programming paradigm, and

(iv) object–oriented programming paradigm.

Today, many expert systems are based on the object–oriented programming paradigm.

3.1.1 Object–oriented programming

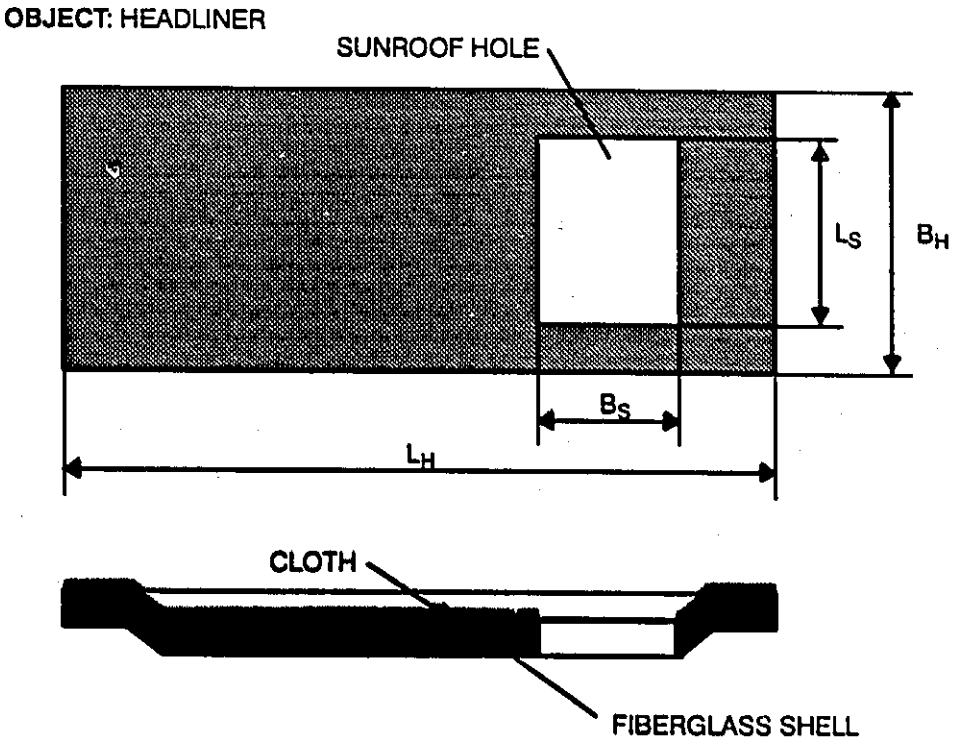
The basic building block in object–oriented programs is the object which is a combination of both data and procedural code in a single structure. Objects can be physical or abstract and represent people, places, things, or ideas in a domain. They have a general form as shown below:

object (<name>, <properties/data>, <behavior/procedural codes>)

The name identifies every object in the program. The data in an object define its properties while the procedural codes define its behavior. This procedural code within an object is called a method and methods are activated by sending messages to objects. Objects are provided with slots which store their properties and behavior. Information stored in a slot is not only specific to the object but is also hidden from other objects and it can be obtained by sending a message to the object.

For example, consider a headliner, an interior trim component for automobiles, as an object. A typical headliner consists of a piece of cloth glued to a fiberglass shell and trimmed to size by a water jet cutting system. Figure 3.2 shows a headliner for an automobile with a sunroof. The model of the automobile in which the headliner is fixed, the color of the cloth, the length and breadth of the headliner, and the length and breadth of the sunroof are the headliner's properties and can be stored in slots. The total perimeter cut and the water jet cutting time can be calculated by procedural codes which can again be stored in slots.

The MODEL and CLOTH.COLOR slots contain string values. The slots



SLOTS	VALUES	VALUE TYPES
MODEL	Automobile model	String
CLOTH.COLOR	Red Chalet	String
HEADLINER.LENGTH	L_H	Number
HEADLINER.BREADTH	B_H	Number
SUNROOF.LENGTH	L_S	Number
SUNROOF.BREADTH	B_S	Number
CALCULATE.CUT.PERIMETER	Procedural code to calculate the total perimeter cut	Method
CALCULATE.CUTTING.TIME	Procedural code to calculate the water jet cutting time	Method

Figure 3.2. Example of an object and its slots

HEADLINER.LENGTH, HEADLINER.BREADTH, SUNROOF.LENGTH, and SUNROOF.BREADTH contain numerical values. Finally, the slots CALCULATE.CUT.PERIMETER and CALCULATE.CUTTING.TIME contain methods. By sending a message to the CALCULATE.CUT.PERIMETER slot, the total perimeter cut on the headliner is determined. Similarly, by sending a message to the CALCULATE.CUTTING.TIME slot, the water jet cutting time which is a function of the total perimeter cut, is also determined.

An object–oriented program which describes a real world application is a collection of objects; for example, each machine in a manufacturing system is an object. All actions in an object–oriented program are performed by sending and receiving messages between various objects. Often, during the development of object–oriented programs we come across objects which have similar properties and behaviors. For example, in a manufacturing application, although each machine is an object, we begin to notice similar objects such as types of lathes or drilling machines. These similar objects can be grouped into classes so once a class is created it serves as an abstraction for creating other similar objects.

A class is a grouping or a generalization of a set of objects. For example, if there are four types of vertical machining centers in a manufacturing system (OKK PCV40 CNC, OKK PCV50 CNC, OKK PCV60 CNC, and OKK PCV620 CNC [40]), then based on their similar properties and behaviors vertical machining centers could be grouped into a class called VERTICAL MACHINING CENTERS as shown in Figure 3.3. Similarly, if there are three horizontal machining centers (OKK PCH400 CNC, OKK PCH500 CNC, and OKK PCH600 CNC), then these horizontal machining centers could be grouped into a class called HORIZONTAL MACHINING CENTERS. It is, however, not necessary to restrict the grouping of objects to two levels only. More than two levels can be created by grouping the

vertical machining centers and horizontal machining centers into a single class called MACHINING CENTERS (Figure 3.4).

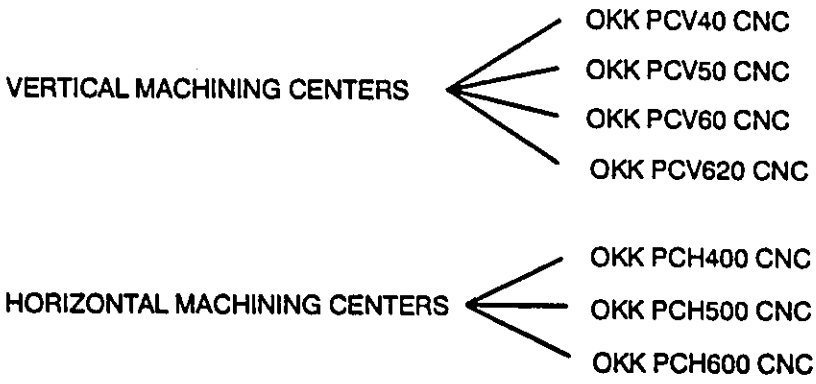


Figure 3.3. Example of grouping objects into classes

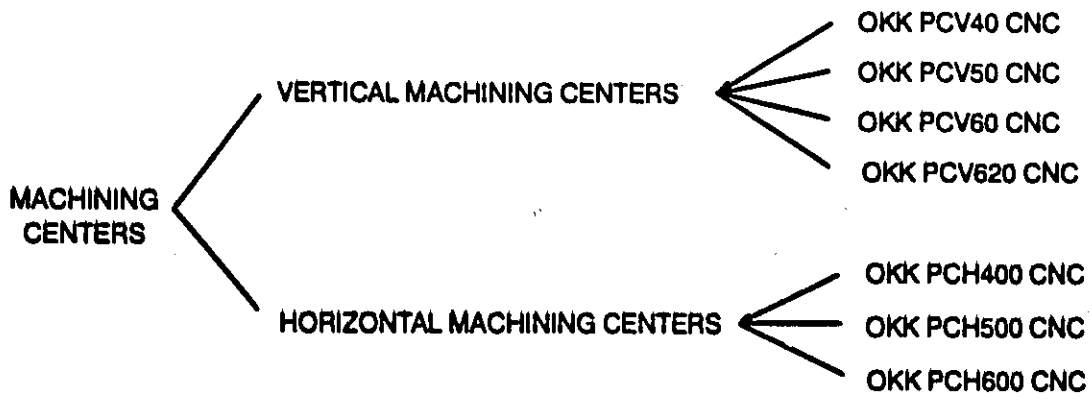


Figure 3.4. Example of further grouping objects into classes

An object-oriented program, therefore, consists of a complex hierarchy of objects in which those having similar properties and behaviors are grouped together. This organization of objects in a tree-like fashion is called an inheritance hierarchy

or a semantic network and it helps conserve information when describing objects. The highest level of objects is at the left side of the network while the lowest level of objects is at the right side and the objects become more specific as the network is traversed from left to right. Except the highest and the lowest level of objects, each object in the network has a parent and a child and in some situations more than one parent or child. For example, the object MACHINING CENTERS has no parents but has two children, HORIZONTAL MACHINING CENTERS and VERTICAL MACHINING CENTERS. On the other hand, the object HORIZONTAL MACHINING CENTERS has one parent, MACHINING CENTERS, and three children, OKK PCH400 CNC, OKK PCH500 CNC, and OKK PCH600 CNC. However, the object OKK PCH600 CNC has one parent, HORIZONTAL MACHINING CENTERS, and no children.

New objects may be created at the appropriate level in the network as children of some objects and parents of some other objects. These new objects would then inherit all the slots and values from their parents, i.e., they inherit their parents' properties and behavior. For example, if the object MACHINING CENTERS has a slot that quantifies its tool magazine capacity and another slot that calculates the cutting speed, then every child of this object inherits these two slots. Therefore, when the HORIZONTAL MACHINING CENTERS object is created as a child object of MACHINING CENTERS, it automatically inherits the slots of its parent as illustrated in Figure 3.5.

Inheritance of slots and values is one of the powerful features of object-oriented programming. Another powerful feature of is that of encapsulation. Encapsulation refers to the fact that a specific type of data and the means to manipulate it can be combined (encapsulated) in a class. These two

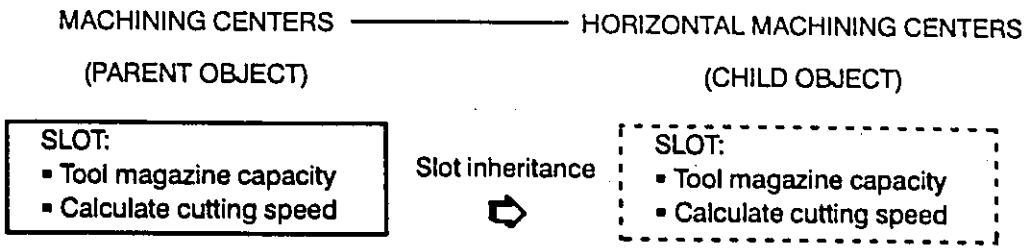


Figure 3.5. Example of slot inheritance between objects

features, inheritance and encapsulation, make object–oriented programs highly reusable and easy to maintain.

3.1.2 Rule–based programming

Humans organize their knowledge about a particular domain in terms of objects, object hierarchies, and object attributes. This, however, provides only descriptive information about the domain. Behavioral knowledge, which can be expressed as rules or methods (Section 3.1.1), is required to solve problems. Rules provide one mode of reasoning while methods provide another.

Rules use knowledge about the dependencies among facts and are expressed in terms of IF...THEN statements. For example, a manufacturing engineer may know the following rule:

IF a slot has to be created in a part

THEN assign the part to a vertical machining center.

The facts on the left hand side of the rule, the IF part of the rule, are called the premises or conditions of the rule. The facts on the right hand side of the rule, the THEN part of the rule, are called the conclusions. Therefore, if all the premises of a rule are true, then its conclusions are also true.

When solving problems in an application domain using heuristics, several rules are often used in succession, starting from facts to intermediate conclusions to more accurate final conclusions. This process of successively using the conclusion of one rule in the premise of another is called chaining. Consider an extension of the previous example with three rules:

Rule 1:

IF a slot has to be created in a part

THEN assign the part to a vertical machining center.

Rule 2:

IF the part is assigned to a vertical machining center
and OKK PCV50 CNC is a type of vertical machining center
and the dimensions of the part are compatible with those of OKK PCV50 CNC

THEN assign the operation of this part to OKK PCV50 CNC.

Rule 3:

IF the operation of this part is assigned to OKK PCV50 CNC

THEN retrieve the process time for this operation.

From the three rules shown above, it is evident that during chaining, the premises of one rule match the conclusions of another rule. Thus, the reasoning portion of an expert system consists of a collection of rules which can be chained in a forward or backward direction. The former method is called forward chaining and the latter one is called backward chaining.

3.1.2.1 Forward chaining

Forward chaining is triggered whenever a new fact is added to the expert system. In this chaining approach, the expert system searches for rules whose premise matches the new fact. If a rule is found and all premises of the rule are true, then the conclusion is asserted and this becomes the new fact. Now the expert system searches for rules whose premise match this new fact. Once a rule is found, then all premises of this new rule are checked to see if they are true. This process continues until no rule with matching premises can be found. An example of a forward chaining strategy, which is also called data–driven reasoning, is shown in Figure 3.6.

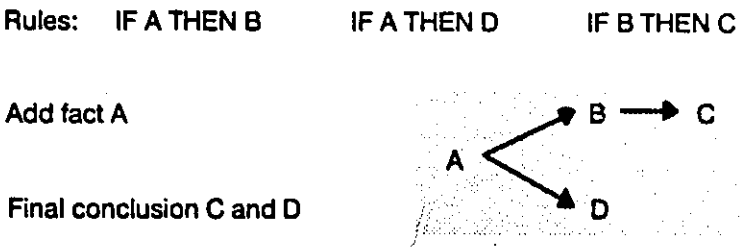


Figure 3.6. Example of a forward chaining process [41]

3.1.2.2 Backward chaining

Backward chaining is triggered whenever a fact or hypothesis has to be verified. In this chaining approach, the expert system searches for rules whose conclusions match the hypothesis to be verified. This hypothesis is considered verified only if all the premises of the rule under consideration can be verified. This may require a search for more rules thereby continuing the backward chaining process. An example of a backward chaining strategy, which is also called

goal–driven reasoning, is shown in Figure 3.7.

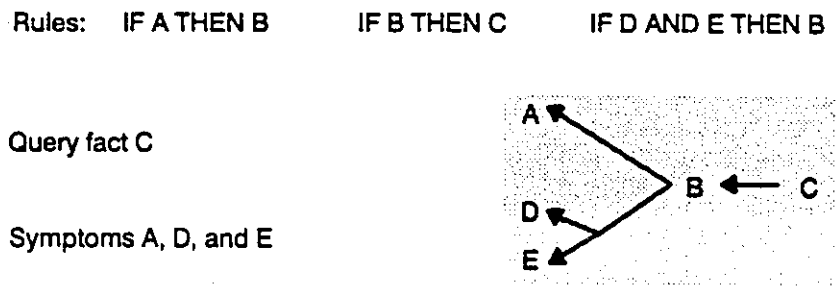


Figure 3.7. Example of a backward chaining process [41]

3.1.3 Knowledge–based simulation systems

Expert systems have always attracted a great deal of interest from the simulation community because of their close similarity with simulation modeling. Both are computer–based models of actual systems, aid in the decision making process, and often interact with other software systems such as a database. O'Keefe [42] has proposed four different ways of combining expert systems and simulation modeling to exploit the similarities between them. These include embedded systems, parallel systems, cooperative systems, and intelligent front or back ends as shown in Figure 3.8.

In a cooperative system the expert system and simulation cooperate on a task and share some data. Cooperative systems can be further subdivided into the following two types:

(i) Type 1 (Figure 3.8.e) in which the expert system and the simulation model share some data but are not embedded in a larger piece of software. This

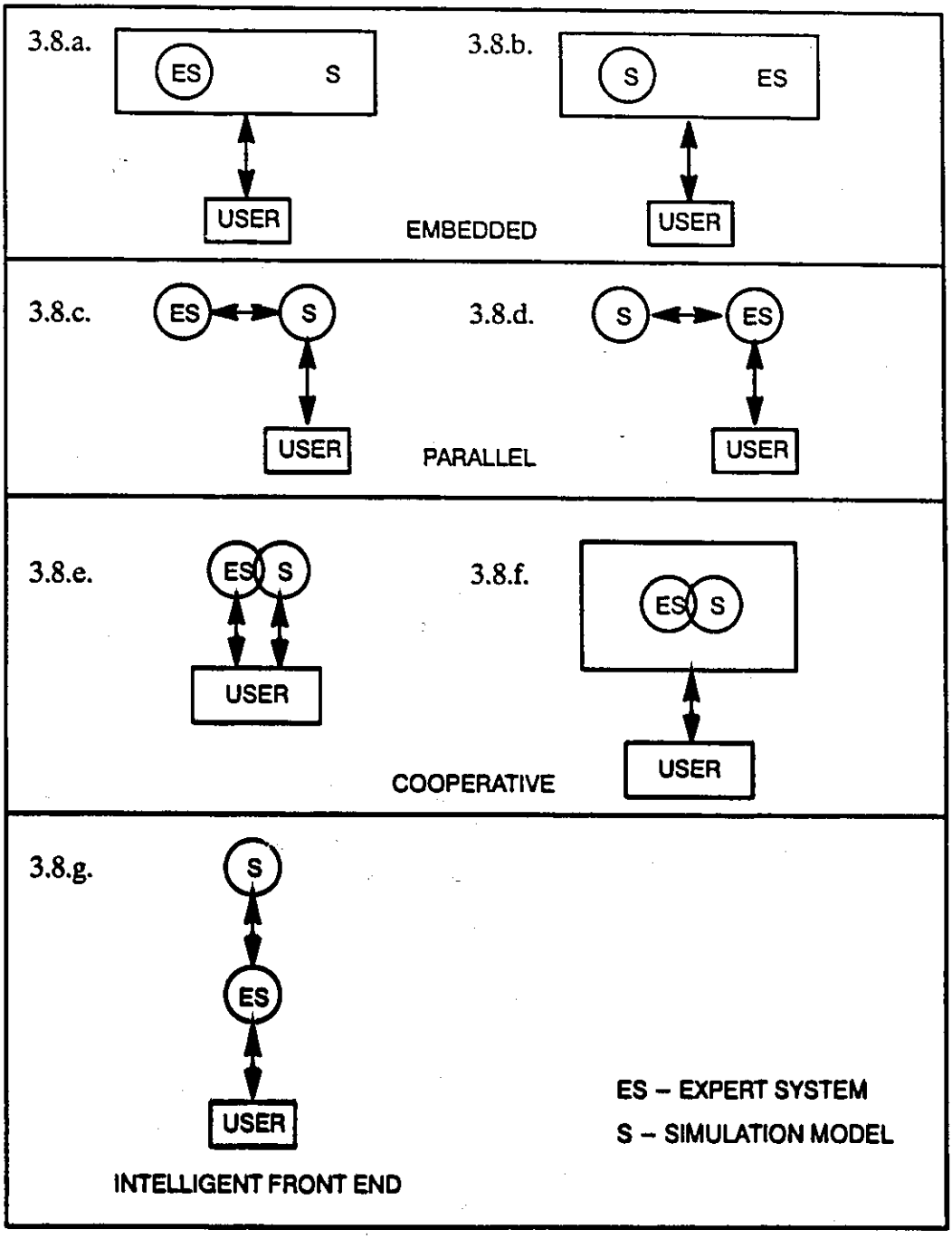


Figure 3.8. Taxonomy for combining experts systems and simulation [42]

results in the expert system and simulation model using different programming paradigms.

(ii) Type 2 (Figure 3.8.f) in which the expert system and the simulation model share some common data and are embedded in a larger piece of software. This results in the expert system and the simulation using the same type of programming paradigm, namely, object–oriented programming.

The second type of cooperative systems was chosen for this research. Knowledge–based simulation systems are a result of this type of cooperative systems.

Knowledge–based simulation systems are the latest generation of simulation software which are used for simulating and analyzing a wide variety of systems. They use both object–oriented and rule–based programming, and are provided with graphics capabilities. Knowledge–based simulation systems have the following advantages:

(i) The various objects in the real world, both physical and abstract, are simulated as objects with properties and behavior. This establishes a close correspondence between the real world and simulated objects. It also helps to represent the real world more accurately and understand the system design easily.

(ii) The inheritance feature of object–oriented programs makes the software reusable and reduces the coding effort by eliminating redundancy in coding.

(iii) The encapsulation feature prevents undesired side–effects from changing the contents of an object because both data and procedures related to an object are stored in one location.

(iii) The rule–based programming paradigm supports the development of rules which reason about the various objects to solve problems.

(iv) The graphics medium provides an excellent interface for the user to communicate with the system.

SIMULATION CRAFT by the Carnegie Group, SimKit by Intellicorp, and LASER/SIM by IntelliSys Corp are some commercially available knowledge–based simulation systems. They are basically shells and can be used by engineers, managers, and supervisors to build knowledge–based simulation models in specific domains.

3.1.3.1 Objects in knowledge–based simulation systems

Knowledge–based simulation systems consist of three different types of objects as shown in Figure 3.9: domain–independent objects, domain–dependent

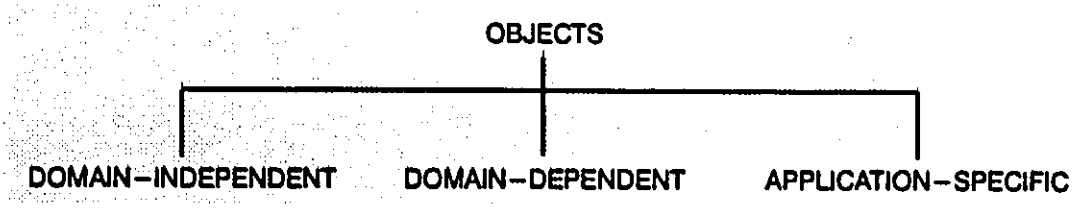


Figure 3.9. Objects in knowledge–based simulation systems

objects, and application–specific objects [43]. Domain–independent objects are objects that are a part of any simulation modeling process, irrespective of the application domain. They model components such as the simulation clock, the simulation calendar, random number generators, and data collectors. Domain–dependent objects, on the other hand, are objects specific to a particular domain and they model components of the actual system. For example, if a manufacturing system is to be simulated, then the domain–dependent objects model components such as machines, material handling systems, and buffers. These

objects act as template for the creation of specific instances. Application–specific objects are certain combinations and numbers of components for a specific analysis. They are instances of both domain–independent and domain–dependent objects; for example, 'workstation–01' and 'workstation–02' are specific instances of a domain dependent object called 'workstation'.

3.1.3.2 SimKit and KEE

The SimKit knowledge–based simulation system was chosen for this research. SimKit is a discrete–event simulation package built on top of the Knowledge Engineering Environment (KEE) expert system shell, as shown in Figure 3.10. KEE is an integrated set of programming facilities that helps software

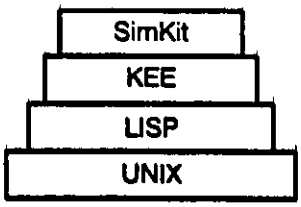


Figure 3.10. The SimKit software environment

developers to build knowledge–based systems in many application domains [44]. KEE is written in Lisp and runs on a SUN workstation under a UNIX operating environment.

KEE is an integrated software system that has a variety of programming tools as shown in Figure 3.11. All these programming tools are centered around the object–oriented representation of KEE which provides a strong foundation for SimKit. By representing components as objects, and their properties and behavior as slots, simulation models can be created quickly and easily. The object–oriented

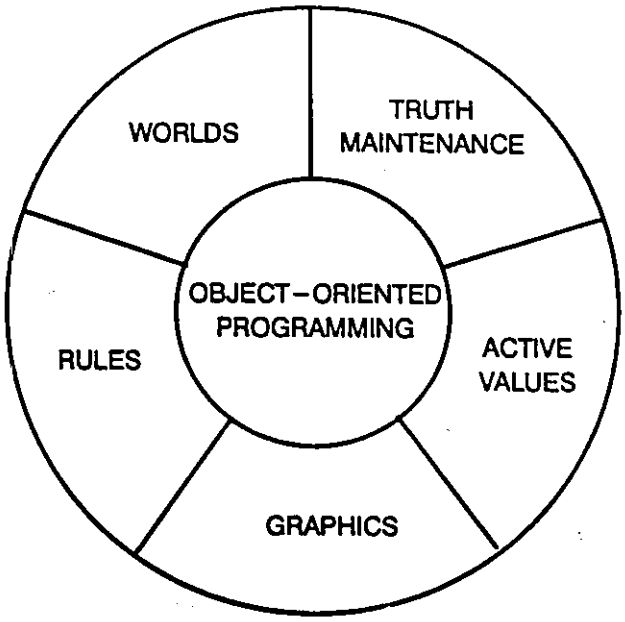


Figure 3.11. Programming tools in KEE [44]

programming paradigm also permits the creation of generic components which can be refined and specialized for specific applications. Rules provide the facility for reasoning within a domain of interest. The worlds programming tool provides facilities for modeling and exploring different hypothetical situations that might arise. Active values provide facilities for causing side–effect behavior when accessing or modifying data in a knowledge base. Truth maintenance is a system for setting up and maintaining dependencies between facts. The graphics interface is an object–oriented tool kit that provides graphics facilities for rapid development of model prototypes.

3.2 The FMX system structure

The main objective of this research is to automate the flexible

manufacturing system design process using expert system techniques and simulation. Flexible manufacturing can be applied to a wide variety of manufacturing processes such as assembly, machining, molding, sheet metal forming, and welding. Since it is impossible to develop a prototype for the entire range of manufacturing processes, it is more beneficial to focus on a specific area of application. The methodology or principles used in this application can then be extended to other areas as well.

In this research, a prototype knowledge-based system, called FMX (Flexible Manufacturing Expert), has been developed for the machining FMS domain. The various modules of FMX, shown in Figure 3.12., include the

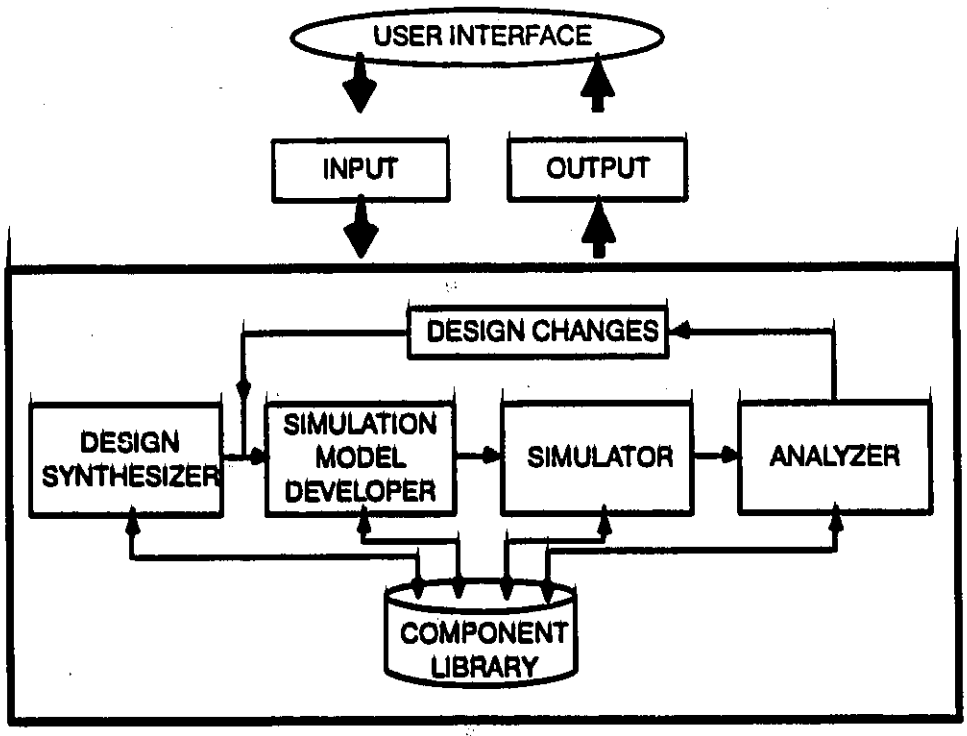


Figure 3.12. Various modules of the FMX prototype

component library, design synthesizer, simulation model developer, simulator, analyzer, and user interface. Furthermore, the component library contains a hierarchical description of both domain independent and domain dependent objects; the design synthesizer contains the heuristic and analytical knowledge in the form of rules and methods to reason about the various objects and generate FMS designs; the design generated by the design synthesizer is converted to a graphical simulation model by the simulation model developer; the simulator then generates output which is analyzed by the analyzer to see if the design meets its performance objectives. If it does not meet the objectives, the analyzer tries to identify the deficiencies and suggests suitable design changes to meet these objectives. This new design is input to the simulation model developer which then generates a simulation model of this design. The user communicates with the various modules through the user interface.

3.3 The Component library

The component library is object–oriented and contains a hierarchical description of both domain–independent and domain–dependent objects. Libraries and models are discussed in the next section followed by the object–oriented modeling of FMS components.

3.3.1 Libraries and models

A library is a knowledge base representing a particular structure of objects, behaviors, and relations from which instances called models can be derived [45]. The object–oriented component library contains both domain–independent and domain–dependent objects. Such a library, however, containing both these object types would be large and hence, difficult to manage. This problem is overcome by

grouping both domain–independent and domain–dependent objects into smaller libraries and using a hierarchical scheme for the libraries.

An example of a hierarchy of three libraries is shown in Figure 3.13. The SIMULATION LIBRARY contains classes of objects relevant to discrete–event simulation such as the simulation clock, the simulation calendar, random number generators, and data collectors. This library is the base library and all other libraries are its sublibraries. The QUEUEING LIBRARY is a sublibrary of the SIMULATION LIBRARY while the SIMULATION LIBRARY is a superlibrary of the QUEUEING LIBRARY. Thus, by having one library on top of another, the sublibrary can inherit information from its superlibrary. The MANUFACTURING LIBRARY is a sublibrary of the QUEUEING LIBRARY and the SIMULATION LIBRARY. Superlibrary and sublibrary are relative terms; a library can be the superlibrary of one library and a sublibrary of another.

A model is a knowledge base representing a particular structure of objects, behaviors, and relations which can be represented and manipulated graphically [45]. Libraries have classes of objects while models have objects that are specific instances and combinations of the objects in all libraries above them. More than one model of a manufacturing system can exist for a specific application. Models always inherit functionalities from libraries which in turn inherit functionalities from their superlibraries. Consider the example shown in Figure 3.14. JOB SHOP MODEL 1 represents one model of a job shop with a certain combination of objects in the MANUFACTURING LIBRARY while JOB SHOP MODEL 2 represents another combination. Both models are members of the MANUFACTURING LIBRARY which in turn is a sublibrary of both the QUEUEING LIBRARY and the SIMULATION LIBRARY. Hence, both these models inherit information from

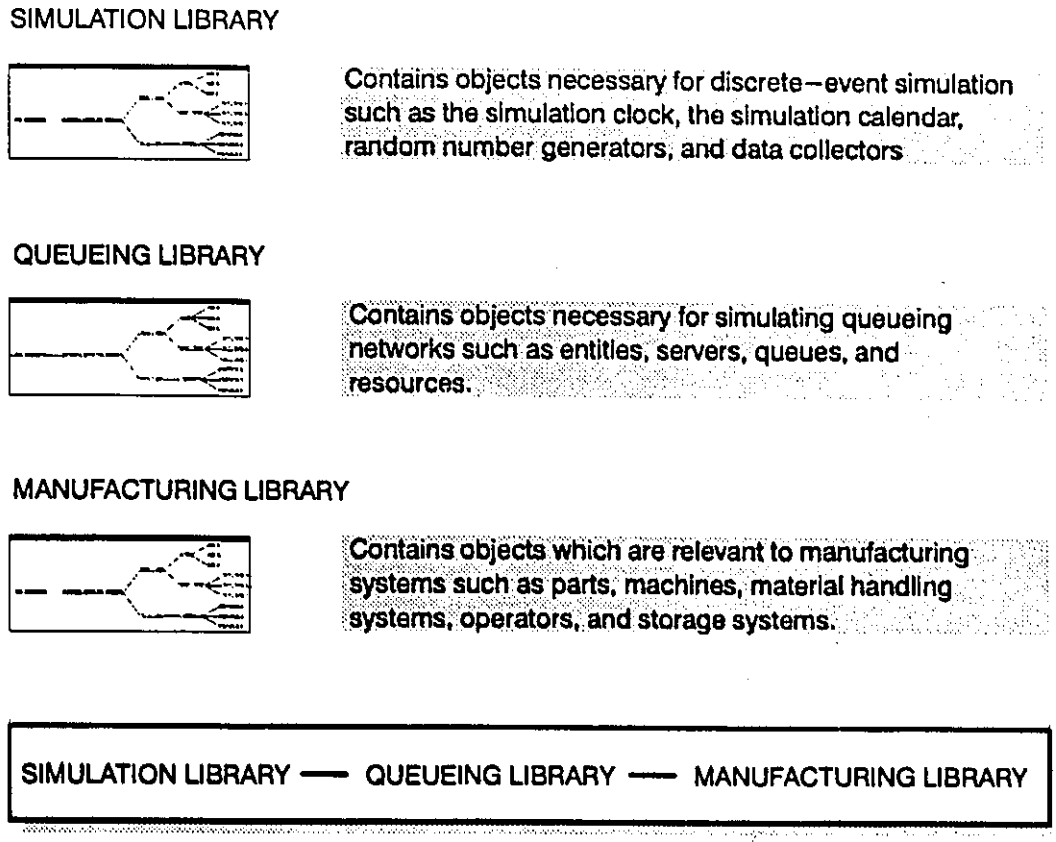


Figure 3.13. A general hierarchy of libraries

Figure 3.14. Hierarchy of libraries and models

all these libraries. Solid lines connect libraries with their sublibraries while dashed lines connect sublibraries with their models (Figure 3.14).

The SimKit knowledge–based simulation system consists of two major

libraries: SIMKIT and QLIB. The SIMKIT library is the top–level library and all other libraries are below it. Therefore, QLIB is a sublibrary of SIMKIT and inherits information from it. Both these libraries contain domain–independent objects and are discussed in the following sections.

3.3.1.1 The SIMKIT library

The SIMKIT library provides all the facilities and behaviors necessary for discrete–event simulation. The two most important mechanisms for scheduling and triggering discrete events, clock and calendar, are provided by this library. An event represents something that takes place at a certain point in time. The time at which it happens is scheduled on a calendar which keeps track of all events and the time they should take place. The clock keeps track of the simulated time and when it reaches the time scheduled for an event, the event takes place. The clock advances in discrete steps from the time of an event to the time of the next event in a simulation calendar.

At each time step, all events to take place are triggered automatically. These events often trigger other events to be executed at the same time or after a specified time delay. New events thus generated are added to the simulation calendar and ordered according to their time of execution. If several events happen at the same time, then the simulation clock is not advanced until all the events scheduled for that time have happened. The SIMKIT library also has the facility to initialize, execute, and trace simulation models.

3.3.1.2 Structure of QLIB library

The QLIB library is a sublibrary of SIMKIT and provides functions necessary for simulating queueing networks such as manufacturing systems, banks,

and communication systems. The simulation model of a queueing network is made up of elements such as entities, servers, queues, resources, sources, and sinks. QLIB is provided with a high-level generic object called QLIB.MODEL.OBJECTS which has four major subclasses as shown in Figure 3.15: QLIB.ITEMS, QLIB.COMPONENTS, QLIB.RESOURCES, and TASKS.

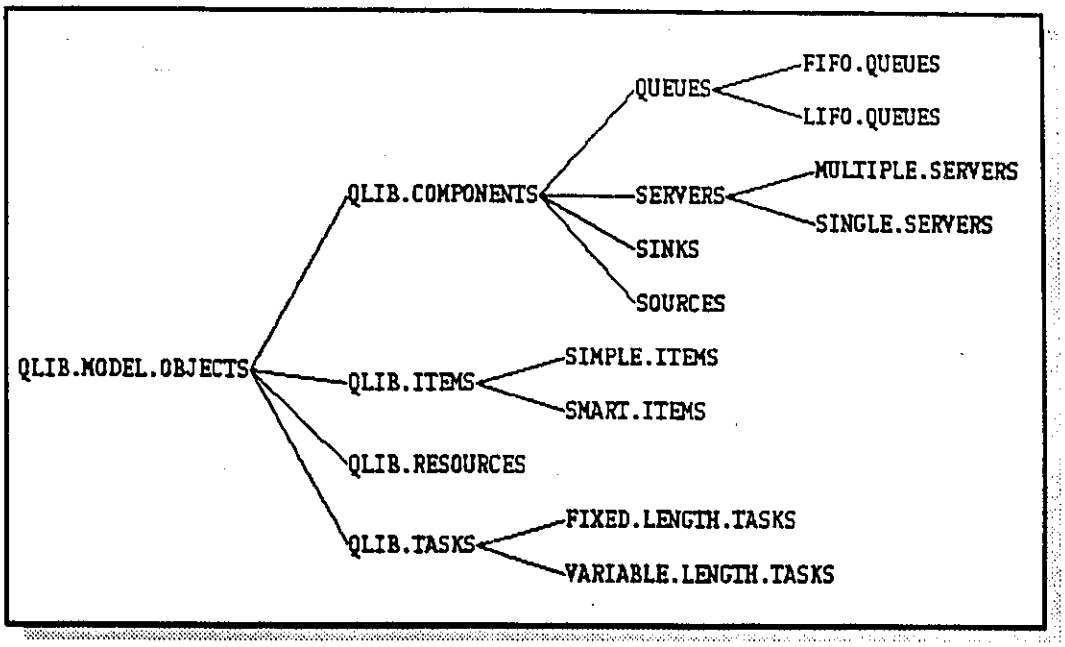


Figure 3.15. QLIB.MODEL.OBJECTS and its subclasses in QLIB

QLIB.ITEMS represent items or entities which are objects that flow through a system, for example, parts in a manufacturing system, customers in a bank, messages in a communication system, etc. There are two types of items: simple items and smart items. Simple items are those items that move from one component to another as they flow through the system undergoing the default activity at each component. Smart items, on the other hand, have a list of tasks that must be completed before they leave the system. When a smart item reaches a

component it checks to see if the component can perform its task. If the component cannot perform the required task, the smart item proceeds to the next component. Both simple and smart items are modeled by the SIMPLE.ITEMS and SMART.ITEMS objects, respectively, and the modeler has the flexibility to model an item as a SIMPLE.ITEM or a SMART.ITEM.

QLIB.COMPONENTS represent components or structural elements in the simulated system and consist of the following types of object: SOURCES, SERVERS, QUEUES, and SINKS. Sources are objects which generate items and are modeled by the SOURCES object. Servers are objects where simulated work gets done, for example, machines in a manufacturing system, tellers in a bank, communication devices in a communication system, etc. When entities arrive at a server they spend a specified amount of time called service time. Therefore, there is always a time delay associated with the servers. Servers are modeled by the SERVERS object. There are two types of servers: single and multiple servers. Single servers are modeled by the SINGLE.SERVERS object and perform tasks on only one item at a time. On the other hand, multiple servers perform tasks on more than one item at a time and are modeled by the MULTIPLE.SERVERS object.

A queue is an object that holds entities waiting to be served by a server. Example of queues are storage racks for parts in a manufacturing system, line-ups at a check-out counter, etc. Queues are modeled by the QUEUES object. There are two types of queues modeled in QLIB based on the queueing discipline: FIFO (first-in-first-out) and LIFO (last-in-first-out) queues. In a FIFO queue, an item that enters the queue first leaves the queue first. On the other hand, in a LIFO queue an item that enters a queue last leaves the queue first. In QLIB, FIFO queues are modeled by the FIFO.QUEUES object and LIFO queues are modeled by the LIFO.QUEUES object. Sinks are objects where the items are removed from

the system by deleting them. They are modeled by the SINKS object in QLIB. Resources, such as pallets or carts in a manufacturing system, are objects for which entities compete. In QLIB, resources are modeled by the QLIB.RESOURCES object.

Tasks are operations that are performed at a component when an item arrives at that component. A component may perform a task on an item or an item may carry out a task at the component. Irrespective of the procedure, the simulation clock is always advanced forward by the number of ticks for a particular task. In QLIB, tasks are modeled by the QLIB.TASKS object. There are two types of tasks: those that take a fixed amount of time and those that take a variable amount of time. These are modeled as FIXED.LENGTH.TASKS and VARIABLE.LENGTH.TASKS, respectively.

3.3.2 Modeling FMS components

The various domain–dependent objects of a machining FMS correlate to components, such as workstations, material handling systems, pallets, and buffers. These were modeled as objects in a new library called FMSLIB. In the hierarchy of the SIMKIT, QLIB, and FMSLIB libraries as shown in Figure 3.16, SIMKIT is

Figure 3.16. Hierarchy of the component library

the superlibrary of both QLIB and FMSLIB, QLIB is a superlibrary of FMSLIB and sublibrary of SIMKIT, and FMSLIB is a sublibrary of both SIMKIT and QLIB

libraries. Hence, objects in FMSLIB inherit information from their parents in the SIMKIT and QLIB libraries.

To develop a simulation model of a flexible manufacturing system design, the system components have to be modeled as entities, servers, queues, or resources. In the current implementation of the component library, FMSLIB, the workstations were modeled as servers, buffers as queues, pallets and material handling systems as resources, and part types as entities. The highest level of objects in FMSLIB that models the components of an FMS in the machining domain is FMSLIB.MODEL.OBJECTS. This object has the following subclasses as shown in Figure 3.17: FMS.ENTITIES, FMS.QUEUES, FMS.RESOURCES,

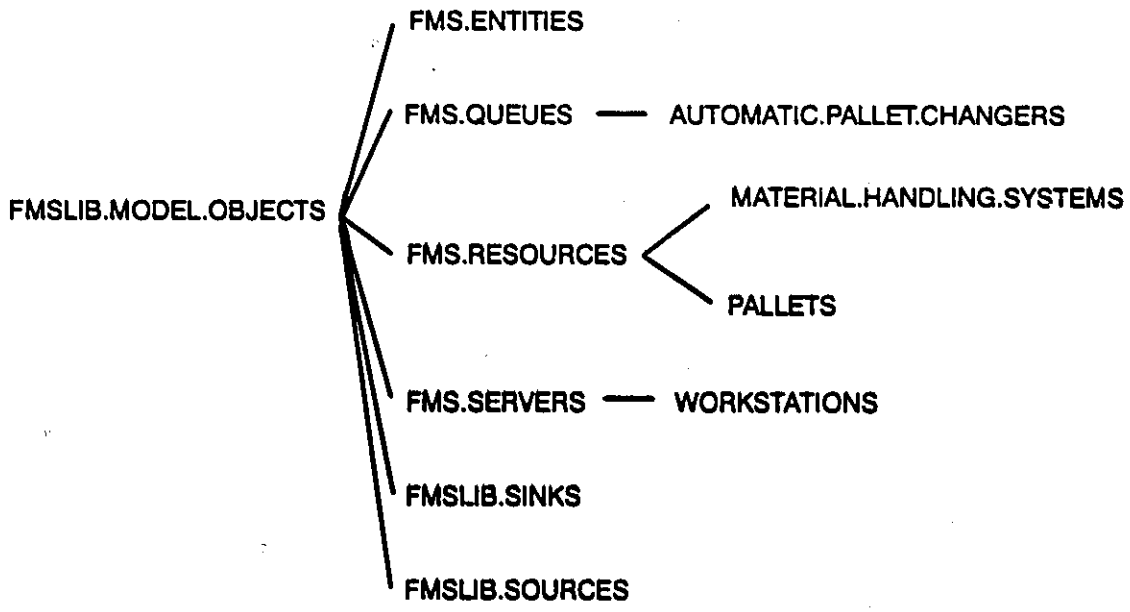


Figure 3.17. Hierarchy of FMSLIB.MODEL.OBJECTS

FMS.SERVERS, FMSLIB.SINKS, and FMSLIB.SOURCES. These subclasses, except FMSLIB.ENTITIES, are further subdivided into other subclasses. The PART.TYPES object which is a subclass of the FMSLIB.ENTITIES object is

modeled in another knowledge base which contains the input information. During consultation, the PART.TYPES object along with its subclasses is copied to the FMSLIB library and made a child of FMSLIB.ENTITIES.

FMS.QUEUES is a class object which is a parent of all automatic pallet changers. FMS.RESOURCES is a class object which is a parent of components that are modeled as resources such as material handling systems and pallets. FMS.SERVERS is a class object which is a parent of all workstations. FMSLIB.SINKS is a class object that models sinks while FMSLIB.SOURCES is a class object that models sources. These objects and their subclasses are discussed in detail in the following sections. An exhaustive list of all components that were modeled and their properties is provided in Appendix A.

3.3.2.1 Workstations

In any manufacturing system, workstations are machines which process parts as they flow through the system. The types of workstations depend on the type of flexible manufacturing system, for example, a machining FMS would consist of numerically–controlled machine tools which can perform metal cutting operations such as turning, milling, drilling, and reaming. On the other hand, a flexible assembly system would consist of robots or special–purpose assembly machines that assemble parts.

In FMSLIB, workstations were modeled as servers and the WORKSTATIONS object, which is a subclass of FMS.SERVERS, is a class object that identifies workstations. The subclasses of the WORKSTATIONS object are LOAD.A, LOAD.B, etc., as shown in Figure 3.18. Several slots describing the various attributes of workstations were attached to the WORKSTATIONS object. These include attributes such as the floor length of a workstation, floor breadth of

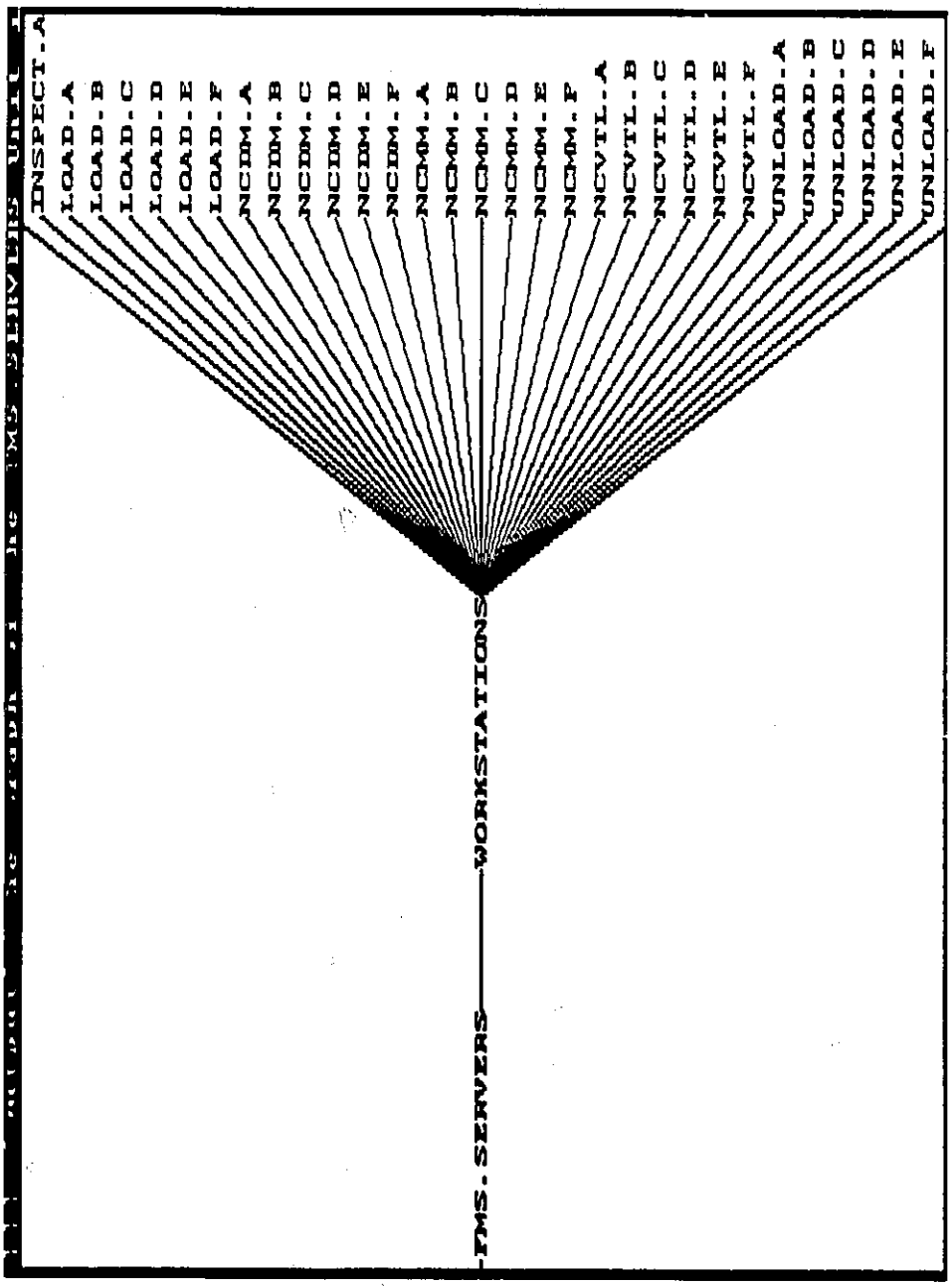


Figure 3.18. The WORKSTATIONS object in FMSLIB

a workstation, maximum combined height of the part and pallet a workstation can accommodate, pallet size the workstation table can hold, minimum clearances required when workstation is positioned next to another workstation, processes that a workstation can perform and the corresponding process times, accuracy that a workstation is capable of delivering, special tools that a workstation can support to perform certain processes, and initial cost of the workstation.

Information contained in these slots are used for selecting appropriate workstations during the design synthesis. These slots are automatically inherited by all children of the WORKSTATIONS object. However, the slot values are provided at the subclass level specific to a particular object. For example, consider the slot PROCESS.CAPABILITY.LIST which contains the list of machining processes a workstation is capable of performing. This slot does not have any value when it is created and attached to the WORKSTATIONS object (Figure 3.19.a). However, a value to this slot is provided for every child of the WORKSTATIONS object later. This is shown in Figure 3.19.b for the NCMM.E object, a child of the WORKSTATIONS object.

Each workstation type was also made a subclass of the SINGLE.SERVERS object which is a subclass of the SERVERS object and so on. The object hierarchy for the NCMM.E workstation type is shown in Figure 3.20. From this hierarchy it can be seen that NCMM.E has multiple parents. Therefore, it inherits slots from all its parents. A few slots inherited by NCMM.E from its parents are shown in Figure 3.21.

Each workstation type is graphically represented by an icon whose dimensions are proportional to the actual length and breadth of its footprints. Icons are associated with objects for the following reasons:

(i) to facilitate the graphical display of the simulation model, and

3.19.a.

```
(Output) The WORKSTATIONS Unit in FMSLIB Knowledge Base
Member slot: PROCESS.CAPABILITY.LIST from WORKSTATIONS
   Inheritance: OVERRIDE.VALUES
   ValueClass: LIST
   Cardinality.Max: 1
   Cardinality.Min: 1
   Comment: "This slot contains the machining process the workstation
            is capable of in the form of a list"
   Values: UNKNOWN
```

3.19.b.

```
(Output) The NCMM.E Unit in FMSLIB Knowledge Base
Member slot: PROCESS.CAPABILITY.LIST from NCMM.E
   Inheritance: OVERRIDE.VALUES
   ValueClass: LIST
   Cardinality.Max: 1
   Cardinality.Min: 1
   Comment: "This slot contains the machining process the workstation
            is capable of in the form of a list"
   Values:
           (MILL-1 MILL-2
                   DRILL-1
                   DRILL-2
                   REAM-1
                   REAM-2)
```

Figure 3.19. Inheritance of slots and values

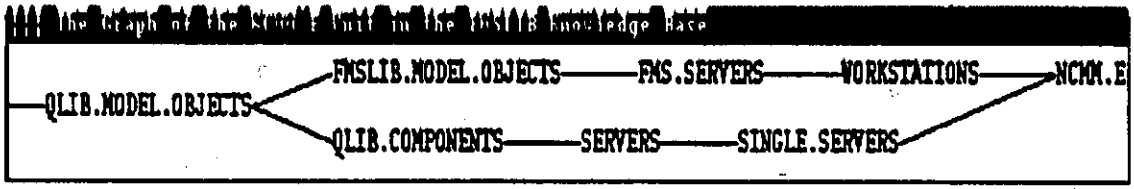


Figure 3.20. Parents of the NCMM.E object

```
Slot                Value                                  Inheritance        Valueclass          Parent object
;;; (Output) The List of the NCMM.E Unit in the FMSLIB Knowledge Base
M: ANYTHING.WAITING?
                    (ANYTHING.WAITING.FOR.SERVER?)         METHOD             (METHOD)            NCMM.E
M: AVERAGE.UTILIZATION
                    UNKNOWN                                OVERRIDE.VALUES    (NUMBER)            WORKSTATIONS
M: CALCULATE.ACTIVITY.COMPLETION.TIME
                    (CALCULATE.ACTIVITY.COMPLETION.TIME)   METHOD             (METHOD)            QLIB.COMPONENTS
M: CALCULATE.ITEM.ARRIVAL.TIME
                    (CALCULATE.ITEM.ARRIVAL.TIME)          METHOD             (METHOD)            QLIB.COMPONENTS
M: CALCULATE.ITEM.DEPARTURE.TIME
                    (CALCULATE.ITEM.DEPARTURE.TIME)        METHOD             (METHOD)            QLIB.COMPONENTS
M: CALCULATE.TIME.UNTIL.START
                    (CALCULATE.TIME.UNTIL.START)           METHOD             (METHOD)            QLIB.COMPONENTS
M: CAPACITY         (1)                                    OVERRIDE.VALUES    (NUMBER)            SINGLE.SERVERS
M: CENTROID.VRT.PICTURE
                    UNKNOWN                                OVERRIDE.VALUES    ((LIST.OF NUMBER))  NCMM.E
```

Figure 3.21. Slot inheritance from multiple parents

(ii) to aid in animation during the simulation.

The icons for three workstation types in the component library are shown in Figure 3.22. From the sizes of these icons it can be seen that NCMM.D is larger

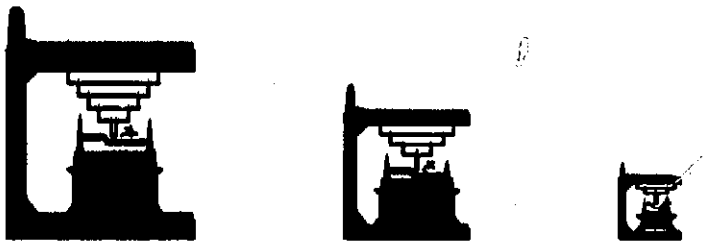


Figure 3.22. Icons for three workstation types

than NCMM.E, which in turn is larger than NCMM.F. Thirty three different types of workstations were modeled in FMSLIB with an icon for each workstation type.

A list of all workstation types, their attributes, values of these attributes, and the icons which represent each workstation type are provided in Appendix A.1.

3.3.2.2 Material handling systems

Material handling systems transport parts and pallets between the workstations. They were modeled as resources in the component library. The following types of material handling systems were considered for this research: automated guided vehicles, rail carts, and conveyors. The MATERIAL.HANDLING.SYSTEMS object, which is a subclass of FMS.RESOURCES, is a class object that identifies material handling systems. The object hierarchy of the MATERIAL.HANDLING.SYSTEMS object is shown in Figure 3.23. The MATERIAL.HANDLING.SYSTEMS object was further

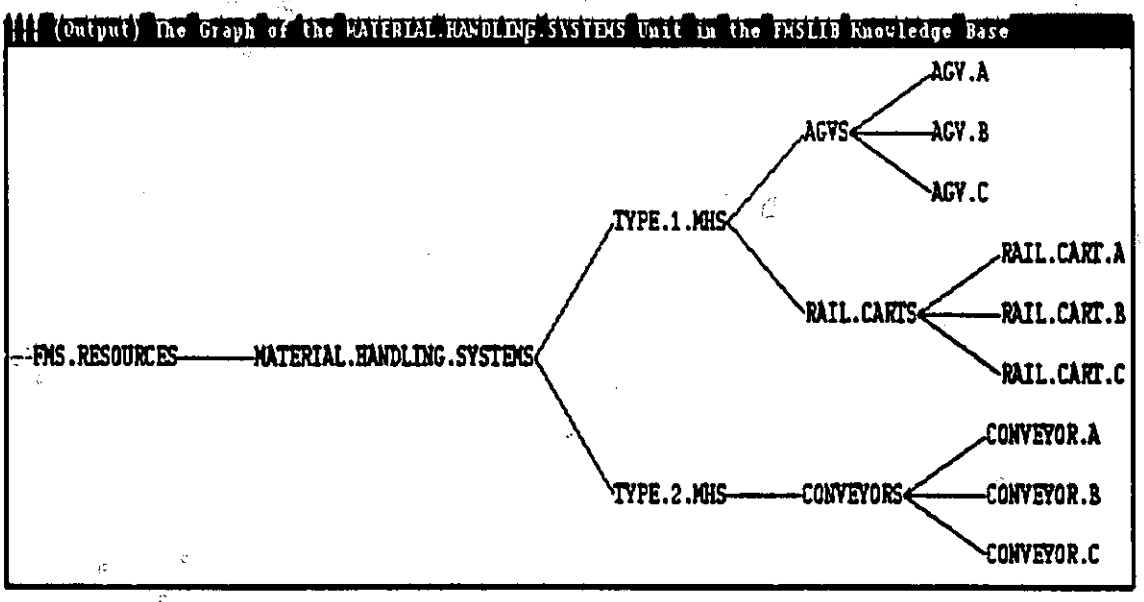


Figure 3.23. The MATERIAL.HANDLING.SYSTEMS object in FMSLIB

subdivided into two types—TYPE.1.MHS and TYPE.2.MHS. This is due to the

differences in attributes and behaviors between conveyors, automated guided vehicles, and rail carts. AGVs and rail carts were modeled as TYPE.1.MHS while the conveyors were modeled as TYPE.2.MHS.

Automated guided vehicles were modeled as unidirectional vehicles while the rail carts were modeled as bidirectional vehicles. The properties of AGVs and rail carts necessary for design synthesis include their length and breadth, their travel speed, their cost, minimum clearance required on either side for travel, dimensions of the pallet they can carry, maximum load they can support, and the distance to be maintained between consecutive tracks. The AGVS object shown in Figure 3.23 is a class object and is the parent of three child objects that were modeled in FMSLIB—AGV.A, AGV.B, and AGV.C. Similarly, the RAIL.CARTS object shown in Figure 3.23 is a class object and is the parent of three child objects that were modeled in FMSLIB—RAIL.CART.A, RAIL.CART.B, and RAIL.CART.C. The icons for the AGV types and the rail cart type are shown in Figure 3.24.

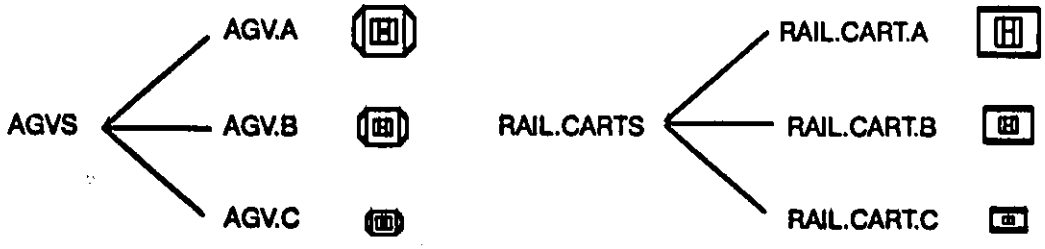


Figure 3.24. Icons for AGVS and RAIL.CARTS objects

The CONVEYORS object, which is shown in Figure 3.23, is a class object and models the various types of conveyors. Properties of the conveyor necessary for design synthesis are the maximum dimensions of the pallet it can carry, cost per unit length, clearance required on either side along its length, travel speed, and

minimum distance between adjacent rows of conveyors. In FMSLIB, three types of conveyors were modeled as child objects of the CONVEYORS object—CONVEYOR.A, CONVEYOR.B, and CONVEYOR.C. Thus, nine different types of material handling systems were modeled in FMSLIB with an icon for each type of material handling system. A list of all material handling system types, their attributes, values of these attributes, and the icons which represent each material handing system type are provided in Appendix A.2.

3.3.2.3 Buffers

In a flexible manufacturing system, it is always essential to keep the part waiting for a workstation or a material handling system than vice versa. The local storage space, or queues, where the parts wait for the workstation or the material handling system are called buffers. Ideally, two types of buffers are needed at each workstation—an input buffer and an output buffer. Input buffer is the storage place where a part waits for the workstation while the output buffer is a storage place where the part waits for the material handling system. A third type of buffer is a common buffer which acts as both input and output buffer. It consists of a rotary table that indexes every time a part has finished a process on the workstation. These are commonly known as automatic pallet changers.

Automatic pallet changers were modeled as queues and the AUTOMATIC.PALLET.CHANGERS object, which is a subclass of FMS.QUEUES, as a class object. The child objects of the AUTOMATIC.PALLET.CHANGERS object are different types of automatic pallet changers, APC.2.A, APC.2.B, APC.2.C, APC.4.A, APC.4.B, APC.4.C, APC.6.A, APC.6.B, and APC.6.C, as shown in Figure 3.25.

Properties of automatic pallet changers which are used during design

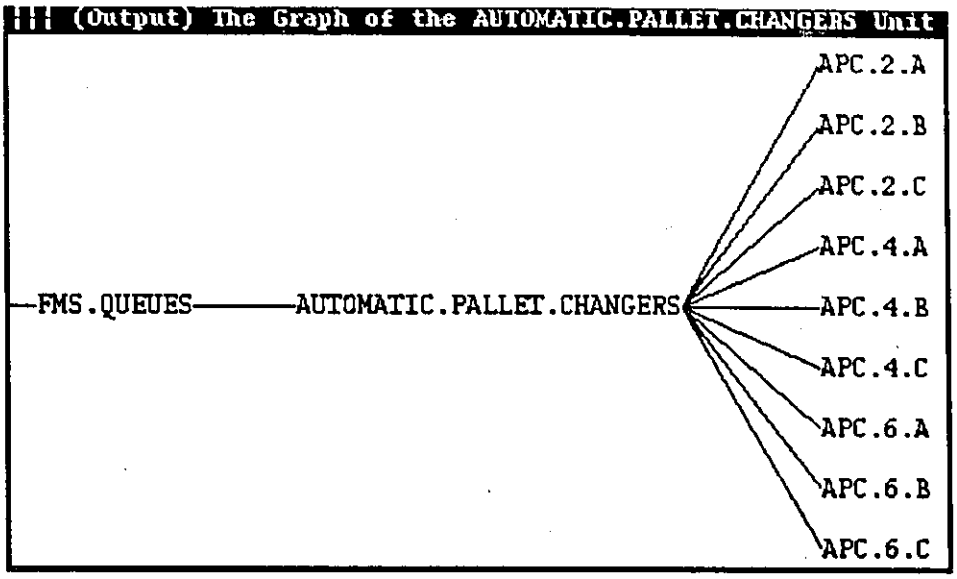


Figure 3.25. The AUTOMATIC.PALLET.CHANGERS object in FMSLIB

synthesis are their length and breadth, initial cost, capacity or the number of pallets they can hold, dimensions of the pallet they can support, and maximum weight they can support. Each automatic pallet changer type was represented by an icon. The icons for three types of automatic pallet changers are shown in Figure 3.26. The

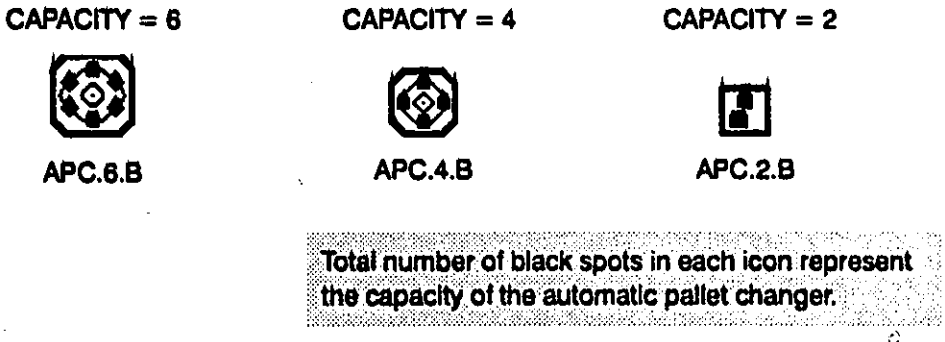


Figure 3.26. Icons for three automatic pallet changers

total number of black spots in each icon represent the capacity of the automatic pallet changer. Nine different types of automatic pallet changers, each with a different capacity, length, breadth, maximum weight carrying capacity, etc., were modeled in FMSLIB with an icon provided for each type. A list of all automatic pallet changers, their attributes, values of these attributes, and the icons which represent each automatic pallet changer type are provided in Appendix A.3.

3.3.2.4 Pallets

Parts are mounted on pallets before processing in an FMS. In FMSLIB, pallets were modeled as resources and the PALLETS object, which is a subclass of FMS.RESOURCES, is a class object that identifies pallets. The properties of pallets used during the design synthesis include their length and breadth, weight, height, and cost.

Three pallet types, PALLET.A, PALLET.B, and PALLET.C, were modeled in FMSLIB. These pallet types can carry only one part and differ from one another in their length, height, weight, and cost. The object hierarchy of the PALLETS object and the icons of each pallet type are shown in Figure 3.27. A list of all pallet types, their attributes, values of these attributes and the icons which represent each pallet type are provided in Appendix A.4.

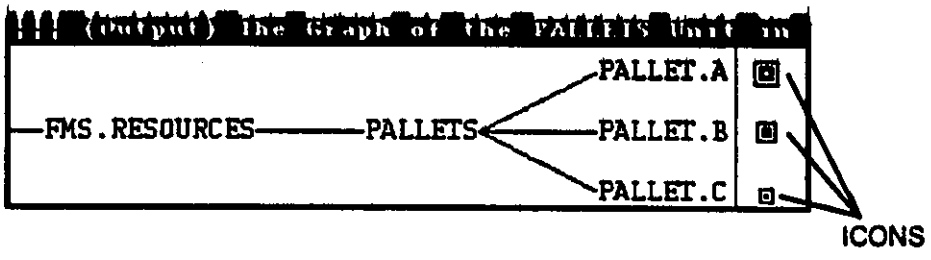
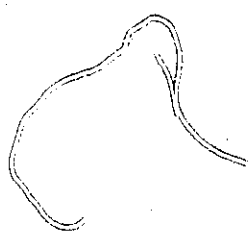


Figure 3.27. The PALLETS object in FMSLIB

3.3.2.5 Sources and sinks

Sources are objects where the part types are created. The PART.TYPE.SOURCES object, which is a subclass of the FMSLIB.SOURCES objects, is a class object which creates the part types. The number of sources depends on the number of part types in the family because each source creates a part type.

Sinks are objects where the part types leave the system. This happens after all the processes of a part type are completed. The PART.TYPE.SINKS object which is a subclass of the FMSLIB.SINKS object, is a class object which deletes the part types. Like the sources, the number of sinks depends on the number of part types where each sink absorbs a part type.



CHAPTER 4

DESIGN SYNTHESIZER

This chapter opens with a brief description of the different types of input information required to drive the design synthesis process. It goes on to describe in detail the FMS design synthesis process for generating an initial FMS design. The heuristic and analytical knowledge for selecting the various system components such as workstations, material handling systems, buffers, and pallets, are then presented. Finally, the implementation of the design synthesizer module of the FMX prototype using KEE is elaborated.

4.1 Input information for design synthesis

Design synthesis is driven by information about the family of part types to be processed and the system requirements. The output is an initial feasible FMS design which is a list of system components. This then serves as the input to the simulation model developer which develops a graphical simulation model of the design. The FMS design synthesis process requires two types of input information as shown in Figure 4.1:

(i) information on the family of part types, and

(ii) information on the system requirements.

These two types of information are described in detail in the following sections.

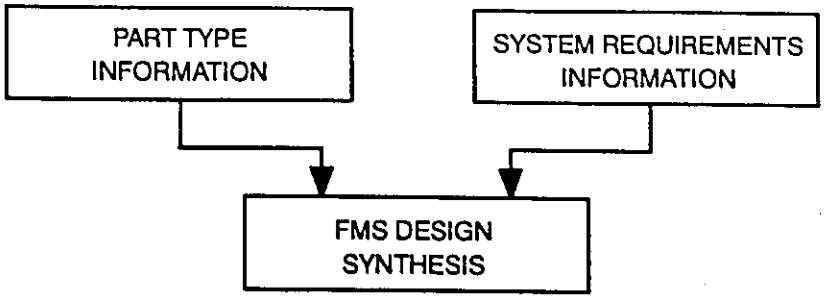


Figure 4.1. Information required for FMS design synthesis

4.1.1 Part type information

It is assumed that the family of part types for which a new flexible manufacturing system is to be designed has already been formed and the process plans for each part type is available. An example of a part type and the information extracted from its process plans are shown in Figure 4.2. This information includes:

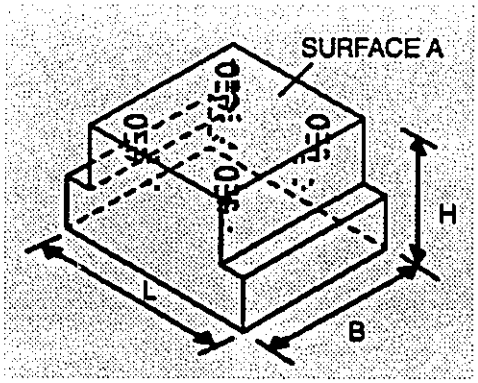
(i) ***Dimensions:*** The overall dimensions of the part type such as its length, breadth, and height.

(ii) ***Weight:*** The weight of the part type.

(iii) ***Process sequence:*** The sequence of machining processes which have to be performed on the part type such as milling, drilling, and turning. These processes are coded; for example, the process of drilling four holes (Φ 25mm) on the part shown in Figure 4.2, is coded as DRILL−1.

(iv) ***Special tools:*** Some processes can be performed using special tools. For example, consider the third process to be performed on the part type shown in Figure 4.2—drill four holes (Φ 25mm) on surface A. This process can be performed by drilling one hole at a time or by drilling all the four holes simultaneously. In the latter situation, however, the machine must be

SAMPLE PART TYPE



Length of part type (L) = 310 mm
Breadth of the part type (B) = 250 mm
Height of the part type (H) = 180 mm
Weight of the part type = 10 kgms
Accuracy required = Low
Production volume = 300 / month

PROCESS SEQUENCE	PROCESS CODE	SPECIAL TOOLS
Load part on pallet	LOAD–1	NO
Mill surface A	MILL–1	NO
Drill four holes (Φ 25mm) on surface A	DRILL–1	YES
Unload part from pallet	UNLOAD–1	NO

Figure 4.2. Example of part type information

equipped with a special tool that can simultaneously drill four holes at the required spacing.

(v) ***Accuracy:*** Accuracy of a part type is a factor which affects the choice of a machine and the one that is selected to perform a process on a part type should be capable of delivering the accuracy required for the part type. Two levels of accuracy, high and low, were considered.

(vi) ***Production volume***: This is the quantity of the part type to be produced within a certain time, for example, 250 units of door panels per month.

4.1.2 System requirements information

Information on the system requirements is used to guide the design synthesis and analysis of simulation output. It includes the following:

(i) ***Floor dimensions:*** This is a measurement of the length and breadth of the available floor space in which the new flexible manufacturing system is to be installed.

(ii) ***Planning horizon:*** This is the time within which the production volumes of each part type are to be met. For example, if a system is to be designed for producing 250 units of PART TYPE A and 400 units of PART TYPE B every month, then the planning horizon is one month.

(iii) ***Acceptable production volume percentage:*** This is the acceptable production volume that should be met for each part type and is expressed as a percentage of the targeted production volume of the part type. For example, the acceptable percentage may be specified as 98% of the targeted production volume of each part type in the family.

(iv) ***Expected utilization:*** Utilization is defined as the percentage of time a workstation or material handling system is busy and is calculated using the relationship in equation (4.1).

$$Utilization = \frac{Busy\ time \times 100}{Total\ simulated\ time}\ \% \qquad (4.1)$$

The expected utilization of a server or resource is specified as a range, for example, the expected utilization of a workstation is specified between 70% and 80%. This workstation is well utilized if the utilization falls between 70% and 80%. However, if the utilization is greater than 80%, then the workstation is over utilized and if the utilization is less than 70%, then the

workstation is under utilized. The expected utilization of each material handling system is also provided as a range. This, however, applies only to automated guided vehicles and rail carts and does not apply to conveyors.

(v) ***Acceptable queue length:*** This is the average capacity that should be available in a buffer or an automatic pallet changer and is expressed as a percentage with respect to the total queue length of each buffer. For example, if the ratio is 75% and total capacity of an automatic pallet changer is four, then there should be at least one empty space in the automatic pallet changer on the average.

(vi) ***Maximum blocking:*** Blocking is a phenomenon that occurs when a part is unable to move on to the automatic pallet changer after being processed by the workstation. This happens when the automatic pallet changer is filled to capacity by parts waiting for the workstation and those waiting for the material handling system. The result is a blocked workstation with a finished part waiting for transportation as shown in Figure 4.3.

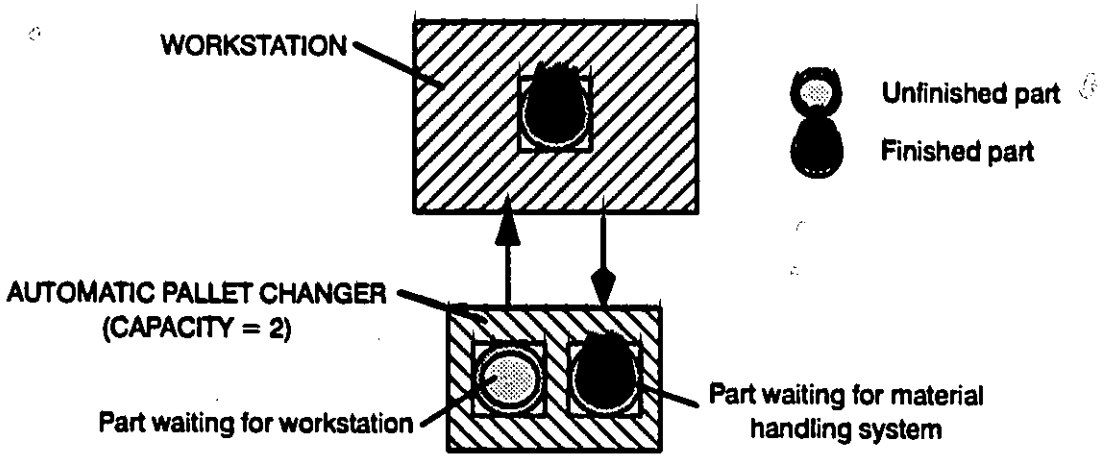


Figure 4.3. Illustration of workstation blocking

Blocking is expressed as a percentage of time a workstation is blocked and is calculated using the relationship in equation (4.2).

$$Blocking = \frac{Blocked\ time \times 100}{Total\ simulated\ time}\ \% \qquad (4.2)$$

(vii) ***System flexibilities:*** Three types of flexibilities were considered in this research [46]: (a) Production volume flexibility—The ability of an FMS to produce part types at different production volumes, (b) Expansion flexibility—The ability to expand the FMS as needed, and (c) Routing flexibility—The ability of routing various part types to different workstations depending on their availability.

Some aspects of the above information guide the design synthesis, or the analysis of simulation output, or both. For example, the length and breadth of the available floor space is used during design synthesis to check if the design generated violates spatial constraints. On the other hand, the information about the minimum production volume to be achieved for each part type is used during the output analysis phase. Information about the expected utilization of workstations is used during design synthesis to determine the number of the workstations and during analysis to check if the actual utilization of each workstation is within the range specified.

4.2 FMS design synthesis

The design synthesis problem is concerned with identifying sufficient system capacity to meet the production requirements. The quality of an FMS design depends largely on the decisions made during the design synthesis phase which involves:

(i) selecting the various types of components and determining their quantities, and

(ii) establishing relationships between these components and configuring them to form a system.

A schematic diagram of the design synthesis process is shown in Figure 4.4.

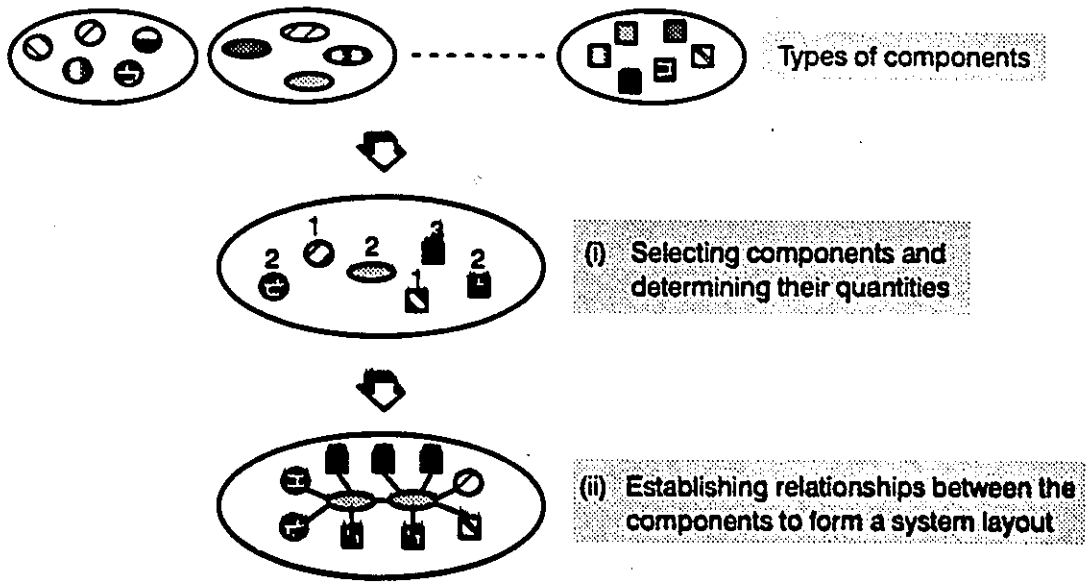


Figure 4.4. Schematic diagram of FMS design synthesis

The design synthesizer module of FMX performs the design synthesis (Figure 3.12). It is an expert system which contains the heuristic and analytical knowledge used by system designers for generating initial FMS designs. The various system components are selected as follows:

(i) The pallet types for each part type in the family are selected first.

(ii) The workstation types are then selected and the quantity of each workstation type is determined.

(iii) The appropriate material handling system is selected and the layout type

is determined.

(iv) Finally, the buffer for each workstation is selected and its capacity determined.

This selection process is illustrated in Figure 4.5.

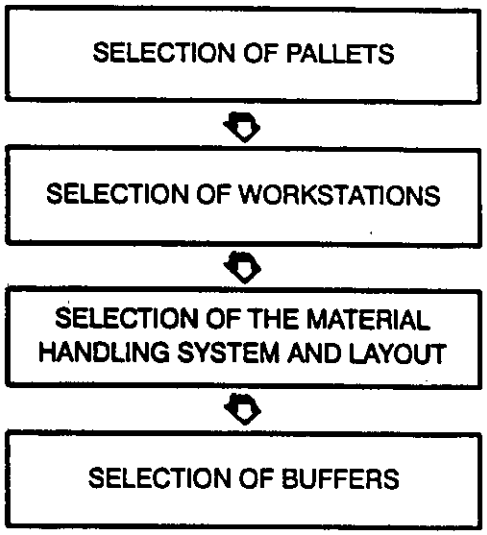


Figure 4.5. Sequential selection of system components

4.2.1 Selection of pallets

Parts are mounted on pallets and are held by fixtures in the proper orientation for processing. In actual practice, there is a pool of pallets with each pallet having a unique identification number and a list of parts it can carry. This list is generated by matching the dimensions of the parts and the pallets. Although in most cases, a pallet carries only one part, there are situations where a pallet can carry more than one part. In this research, it is assumed that a pallet can carry only one part.

The selection of the right pallet type for each part type in the family

depends on the dimensions the pallet type and the part type. Any pallet type whose dimensions are greater than those of a part type is a feasible pallet type to carry that part type. Consider a part type and pallet type as shown in Figure 4.6. The pallet type is feasible to carry the part type if P_L and P_B are both greater than D_P. A feasible set of pallet types is thus generated for each part type and the smallest pallet type is then selected from this feasible set to carry the part type. Examples of rules for the selection of pallets are:

IF dimensions of a part type are less than the dimensions of a pallet type

THEN the pallet type is a feasible pallet type to carry the part type.

IF the feasible set of pallet types to carry a part type is more than one

THEN select the smallest pallet type to carry the part type.

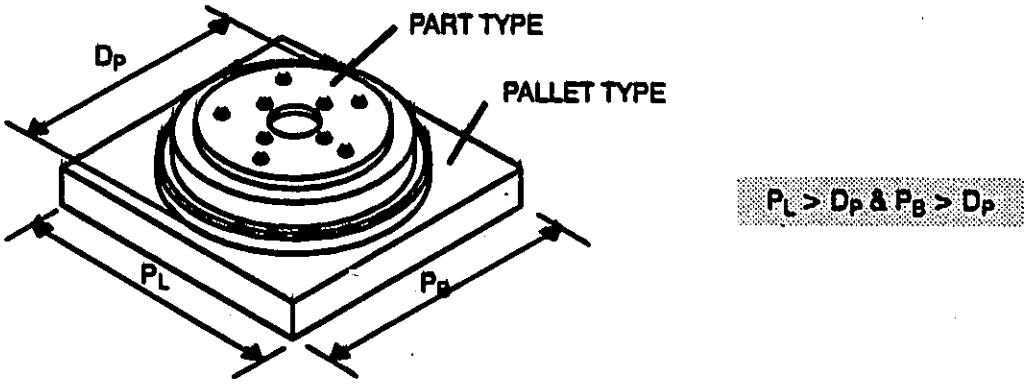


Figure 4.6. Feasible pallet type for a part type

The quantity of each pallet type depends on the flow time of the part types they carry. The flow time of each part type, however, is calculated only after the

workstations are selected and the routing of each part type is determined. After a pallet type has been selected to carry a part type, the combined weight and height of both the part type and pallet type are calculated. These values are used for selecting the workstations and buffers.

4.2.2 Selection of workstations

The selection of a group of workstations to process a family of part types depends on physical, technical, and economic factors. Workstation types are selected by matching the attributes of the part types with corresponding attributes of a workstation type. A feasible set of workstation types is generated for each process in the process sequence of a part type. A feasible workstation type in this feasible set is the one that satisfies the following constraints:

(i) It is capable of performing the process.

(ii) It can accommodate the part type and the pallet type on which the part type is mounted. A workstation type can accommodate a part type and the pallet type on which it is mounted when: a) the dimensions of its table are greater than or equal to the dimensions of the pallet type, and b) its work space height is greater than or equal to the combined height of the part type and the pallet type.

(iii) It can support the combined weight of the part type and the pallet type.

(iv) It can deliver the same accuracy as that required by the part type.

(v) It has special tools if the process requires special tools.

When this feasible set is generated for a process of a part type, the corresponding process times are also retrieved simultaneously.

The feasible set of workstation types for each process of a part type and the process times are expanded into a larger list for the part type. For example,

consider a family of two part types—part type 1 and part type 2. Let the process sequence of both part types consist of three machining processes and A, B, C, and D denote four workstation types that are available. Let the resulting feasible set of workstation types and process times be as follows:

Part type 1 – [(A) (B C) (A D)] & [(t_1) (t_2 t_3) (t_3 t_4)]

Part type 2 – [(A) (C D) (B)] & [(t_2) (t_1 t_3) (t_5)]

The first process of part type 1 could be performed by workstation type A in t_1 time units while the second process could be performed by workstation type B or C in t_2 or t_3 time units, respectively. Similarly, the first process of part type 2 could be performed by workstation type A in t_2 time units while the second process could be performed by workstation type C or D in t_1 or t_3 time units, respectively.

This feasible set of workstation types for each part type is further decomposed to generate a set of alternatives where each alternative indicates a combination of workstation types to process a part type. For example, Figure 4.7 shows the alternatives generated for the two part types. If more than one alternative

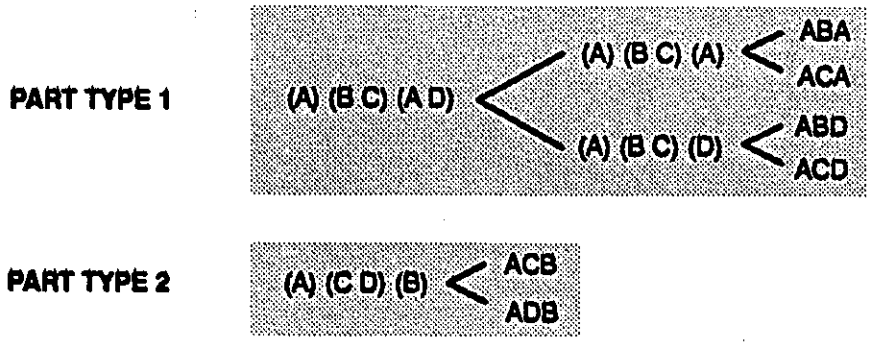


Figure 4.7. Alternatives for each part type

exists for a part type, then the investment cost for each alternative is determined and the alternative that results in the lowest cost is chosen.

To calculate the investment cost for each alternative, the number of workstations of each type is calculated first using equation (4.3). The total cost of the workstations for each alternative is then calculated using equation (4.4). The best alternative for each part type is finally determined using equation (4.5).

$$NW_{ijl} = \left\lceil \frac{\left[\sum_{k=1}^{k=P_i} O_{ijkl} * t_{ijkl} \right] * V_i}{H * U_j} \right\rceil \tag{4.3}$$

$$CA_{il} = \sum_{j=1}^{j=W} C_j * NW_{ijl} \tag{4.4}$$

$$BA_i = \min \left\{ CA_{i1}, CA_{i2}, ..., CA_{il}, ..., CA_{iTA_i} \right\} \tag{4.5}$$

where,

N	=	Total number of part types in the family
W	=	Total number of workstation types
H	=	Planning horizon
P_i	=	Total number of processes in the process sequence of part type i
V_i	=	Production volume of part type i
BA_i	=	Best alternative for part type i
TA_i	=	Total number of alternatives for part type i
U_j	=	Expected utilization of workstation j
CA_{il}	=	Total cost for alternative l of part type i

O_{ijkl} = 1 if process k of alternative l of part type i is done on workstation type j

0 if process k of alternative l of part type i is not done on workstation type j

t_{ijkl} = Process time for process k of alternative l of part type i on workstation type j

NW_{ijl} = Number of workstation type j required for part type i and alternative l

$(i = 1,2,...,N)$

$(j = 1,2,...,W)$

$(k = 1,2,...,P_i)$

$(l = 1,2,...,TA_i)$

The workstation types of the best alternative for each part type are then pooled together and the entire set of workstation types for processing the family are determined. Consider the example shown in Figure 4.7. Let the alternative chosen for processing part type 1 be ABA and the alternative for part type 2 be ADB. The workstation types for the entire family after pooling are A, D, and B. The quantity of each workstation type is calculated using equation (4.6) which is a variation of equation (4.3).

$$NW_j = \left\lceil \frac{\left[\sum_{i=1}^{i=N} \left[\sum_{k=1}^{k=P_i} O_{ijk} * t_{ijk}\right] * V_i\right]}{H * U_j} \right\rceil \qquad (4.6)$$

where,

O_{ijk} = 1 if process k of part type i is done on workstation type j
0 if process k of part type i is not done on workstation type j

t_{ijk} = Process time for process k of part type on workstation type j

NW_j = Number of workstation type j required for all part types

After the quantity of each workstation type is calculated, the routing of each part type is generated. Consider the example shown in Figure 4.7. Let the quantity of each workstation type calculated using equation (4.6) be the following: three for A (A.1, A.2, and A.3), two for B (B.1 and B.2), and one for D (D.1). The routes for each part type are as follows:

Part type 1 – (A.1 A.2 A.3) (B.1 B.2) (A.1 A.2 A.3)

Part type 2 – (A.1 A.2 A.3) (D.1) (B.1 B.2)

This means that the first process of part type 1 could be performed by any one of A.1, A.2, or A.3, the second by any one of B.1 or B.2, and the third by any one of A.1, A.2, or A.3. The workstation to which each part type would be sent depends on the availability of a workstation during the simulation.

Flow time is defined as the sum of the process times in the process sequence of a part type, the travel times, and the waiting times. However, at this point in the design process, the travel and waiting times are unknown. Therefore, the flow time for each part type is calculated using the process time as shown equation (4.7). From this flow time, the minimum number of pallet types required to meet the production volume of each part type is calculated using equation (4.8).

$$FT_i = \sum_{k=1}^{k=P_i} t_{ik} \qquad (4.7)$$

$$\textit{Minimum units of pallet type for part type } i = \frac{FT_i * V_i}{H} \qquad (4.8)$$

where,

FT_i = Flow time for part type i

V_i = Production volume of part type i

H = Planning horizon

Example of some rules for selecting the workstation types and determining their quantity are shown below:

IF a workstation type can perform a process in the process sequence of a part type
and the workstation type can accommodate both the part type and the pallet type on which it is mounted
and the workstation type can support the load of both the part type and the pallet type
and the workstation type can deliver the same accuracy as that required by the part type
and the workstation type has special tools in it which are needed by this process of the part type

THEN the workstation type is a feasible workstation type for processing the part type.

IF more than one workstation type can perform a process in the process sequence of a part type

THEN call method to generate a list of alternatives to process the part type and call method to find the alternative that results in the least cost.

4.2.3 Selection of material handling system and layout

Material handling system selection and layout design are inter-related problems and the choice of one influences the other. There are two approaches to resolve this issue:

(i) to assume that the layout is known and then decide the appropriate material handling system [47], or

(ii) to assume that the material handling system is known and then design the layout [48].

The chosen approach depends on the type of manufacturing system. The former is chosen when the manufacturing system is conventional and the material handling system is not automated. The latter is chosen when the manufacturing system is automated and so is the material handling system. An FMS is automated and hence, the material handling system is selected before deciding the system layout.

The type of material handling system such as AGVs, rail carts, or conveyors depends on the flexibility expected from the system. The following flexibilities [46] were considered in this research:

(i) Production volume flexibility – The ability of an FMS to produce part types at different production volumes.

(ii) Expansion flexibility – The capability of expanding an FMS as needed.

(iii) Routing flexibility – The ability of routing various part types to different workstations depending on their availability.

The material handling system is selected first based on the flexibility expected from the system and then the layout is selected. Example of rules for selecting the type of material handling system based on these flexibilities are shown below:

IF the routing flexibility is high
and the expansion flexibility is high
and the production volume flexibility is high
THEN select AGV as the material handling system.

IF the routing flexibility is medium
and the expansion flexibility is low
and the production volume flexibility is low
THEN select conveyors as the material handling system.

IF the routing flexibility is medium
and the expansion flexibility is medium
and the production volume flexibility is medium
THEN select rail carts as the material handling system.

The layout of any manufacturing system is the spatial arrangement of its components, such as workstations, material handling system, and buffers. It is a top view of the system which shows the location of each component relative to other components in the system. In an automated manufacturing system, such as an FMS, the layout depends on the type of material handling system and the work flow.

Work flow in an FMS is the flow of parts and is classified into serial work flow and parallel work flow. A serial work flow results when each part type in the family passes through a set of workstations sequentially. A parallel work flow results

when at least one part type in a family returns to the loading/unloading station for refixturing or reorienting before the next set of machining processes. For example, let a system process a family of three part types and consist of a loading station (L) and unloading station (U), and three workstations—A, B, and C. The work flow of the family of part types is determined from the routing of each part type. Figure 4.8.a and Figure 4.8.b are examples of serial work flow while Figure 4.8.c is an example of a parallel work flow.

4.8.a.

PART TYPE 1	L–A–B–C–U
PART TYPE 2	L–A–B–C–U
PART TYPE 3	L–A–B–C–U

4.8.b.

PART TYPE 1	L–A–B–C–U
PART TYPE 2	L–A–B–U
PART TYPE 3	L–A–C–U

4.8.c.

PART TYPE 1	L–A–B–L–C–U
PART TYPE 2	L–C–L–A–L–B–U
PART TYPE 3	L–A–C–L–A–U

Figure 4.8. Various types of work flow

According to Ohmi [50], there are four basic layout patterns as shown in Figure 4.9: linear, loop, net, and tree. Of these, only the linear and loop layout were considered in this research.

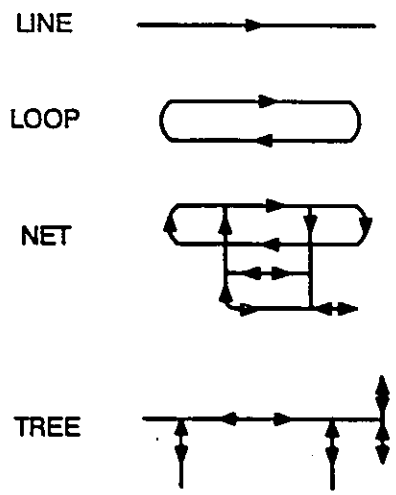


Figure 4.9. Various types of layout

Example of some rules for selecting the type of layout are shown below:

IF all the part types in the family have identical routing
or the routing of some part types are subsets of the routing of other part types
and no part types return to the loading station for refixturing or reorienting
THEN the work flow of the family is serial.

IF each part type in the family has a different route
or at least one part type returns to the loading station for reorienting or refixturing
THEN the work flow of the family is parallel.

IF the type of flow is parallel
and rail carts have been selected as the material handling system
THEN select a line type layout.

IF the type of flow is parallel
and AGV or conveyors have been selected as the material handling system
THEN select a loop type layout.

IF the type of flow is serial
and AGV or conveyors have been selected as the material handling system
THEN select a loop type layout.

After selecting the layout, the positions of the workstations are determined. For both line and loop type layout, the workstations are positioned on either side of the line or the loop as shown in Figure 4.10. When workstations are positioned

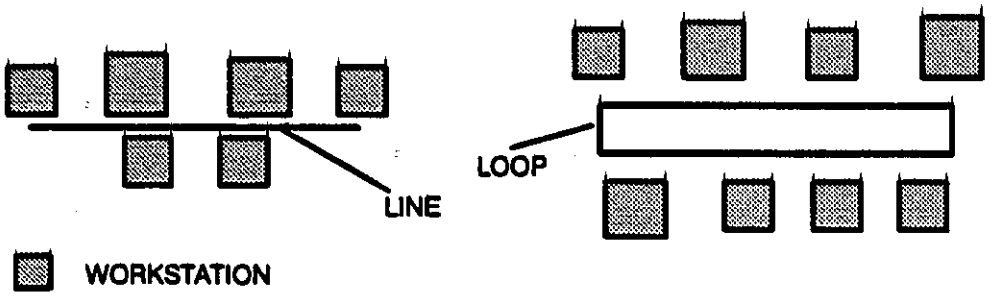
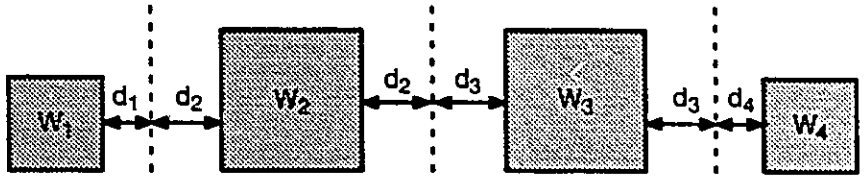


Figure 4.10. Positioning the workstations in a layout

beside one another in the system, the clearances required on either side of these workstations must be considered as shown in Figure 4.11.



W_i _ Workstation i

d_i – Clearance for workstation i

Figure 4.11. Example of workstation clearances

Given a certain number workstations, a work flow type, and a layout type, there are several combinations of workstation arrangements that could be generated. Since the objective is to select the layout that minimizes travel time, the combination of workstations on either side of the layout which results in the minimum travel distance and hence minimum travel time is selected.

For example, consider a situation where the system consists of seven workstations, a serial work flow and a loop layout. There are several ways of positioning these workstations on either side of the loop. One extreme is the case where all seven workstations are positioned on the top of the loop. The other extreme is the case where all seven workstations are positioned at the bottom of the loop. In between these two extremes is a combination of workstations that minimizes the travel distance. Appendix B illustrates the various workstation positions in a loop layout. From this combination, the workstations that are placed on the top of loop and those that are placed on the bottom are determined. This

information is later used by the simulation model developer when developing a graphical simulation model of the design.

After selecting the type of material handling system such as an AGV, rail cart, or conveyor, the next step is to select a specific type of AGV or rail cart or conveyor. For example, if AGVs are selected as the material handling system, then the following step is to select AGV.A, AGV.B, or AGV.C. The selection of a type of AGV or rail cart is different from that of the conveyor because of their modes of operation.

For the AGVs and rail carts, a feasible set of rail cart types or AGV types is generated. This feasible set contains those types of rail carts or AGVs which can:

(i) support the heaviest combined weight of a part type and pallet type circulating in the system, and

(ii) accommodate the size of the largest pallet type circulating in the system.

The minimum number that is required for each type of AGV or rail cart in this feasible set is then calculated. The type of AGV or rail cart which results in the lowest cost is then selected.

For conveyors, the type of conveyors that can accommodate the largest pallet type circulating in the system form the feasible set. From this set, the type of conveyor which results in the lowest cost is then selected. Example of some rules for selecting the type of AGV, rail cart, or conveyor are shown below:

IF AGVs are the selected material handling system
and the maximum weight to be carried is <= the load capacity of of a certain type of AGV
and the maximum size of pallet type used is <= the maximum pallet size that the type of AGV could carry

THEN this type of AGV is a feasible type.

IF conveyors are selected as the material handling system
and the maximum size of pallet type used is <= the maximum pallet size that a certain type of conveyor could carry
THEN this type of conveyor is a feasible type.

IF AGVs are selected as the material handling system
and the feasible set contains more than one type of AGV
THEN call method to calculate the quantity of each type of AGV in this feasible set
and call method to select the AGV type that results in the lowest cost.

The initial speed at which a selected material handling system travels is set at the average value of the lower and upper limits of the speed range.

4.2.4 Selection of buffers

Automatic pallet changers act as storage spaces or buffers for parts at a workstation. Each workstation is provided with an automatic pallet changer. The purpose of an automatic pallet changer is to keep the part waiting for a workstation or a material handling system rather than vice versa. The selection of a suitable automatic pallet changer for each workstation depends on the total number of pallets circulating in the system, the largest pallet type visiting the workstation, and the weight supported by the automatic pallet changer when it is loaded to capacity.

The capacity of the selected automatic pallet changer is the minimum that is required. Example of some rules for selecting the automatic pallet changers are:

IF the ratio of the total number of pallets to the total number of workstations is greater than 0 and less than or equal to 1
THEN each workstation should have an automatic pallet changer with a capacity of 2.

IF an automatic pallet changer has the required capacity
and the pallet size it can hold is equal to the maximum pallet size that visits the workstation
and it can support the heaviest combination of part and pallet when loaded to capacity
THEN select this automatic pallet changer as the buffer for the workstation.

4.3 Implementation of the design synthesizer

The design synthesizer contains both heuristic and analytical knowledge that is necessary for generating an initial FMS design. In knowledge-based systems heuristic knowledge is represented as rules while analytical knowledge is represented as methods.

4.3.1 TellAndAsk language

In KEE, rules are represented in the following format:

(IF <PREMISES> THEN <CONCLUSIONS>)

It is also possible to have rules with multiple premises and conclusions as shown below:

(IF <PREMISE 1> AND <PREMISE 2> ..... <PREMISE N>

THEN <CONCLUSION 1> AND ..... <CONCLUSION M>)

When these rules are chained, in a forward or backward direction, the premises of one rule match the conclusion of other rules.

Rules reason about objects in knowledge bases and hence, have to refer to knowledge bases constantly. Because the TellAndAsk language of KEE, by virtue of its English–like syntax refers to objects in knowledge bases, rules are written using this language [51]. Examples of two rules written using the TellAndAsk language are shown below:

```
(DRILLING.MACHINE.RULE
   (IF   (the CURRENT.PROCESS of PART.A is DRILLING)
         (the PROCESS.CAPABLE of MACHINE.A is DRILLING)
         THEN
         (the MACHINE.TO.SEND.TO of PART.A is MACHINE.A)))

(MILLING.MACHINE.RULE
   (IF   (the CURRENT.PROCESS of PART.A is MILLING)
         (the PROCESS.CAPABLE of MACHINE.B is MILLING)
         THEN
         (the MACHINE.TO.SEND.TO of PART.A is MACHINE.B)))
```

These two rules are translated as follows: if the CURRENT.PROCESS slot of PART.A object has the value DRILLING, and if the PROCESS.CAPABLE slot of MACHINE.A object has the value DRILLING, then the MACHINE.TO.SEND.TO slot of PART.A object will get the value MACHINE.A.

Similarly, if the CURRENT.PROCESS slot of PART.A object has the value MILLING, and if the PROCESS.CAPABLE slot of MACHINE.B object has the value MILLING, then the MACHINE.TO.SEND.TO slot of PART.A object will get the value MACHINE.B. The syntax of these rules show how various portions of a rule refer to objects and their slots in a knowledge base.

4.3.1.1 Variables in rules

The rules mentioned earlier in Section 4.3.1 are similar in that they express similar facts. The premises and conclusions of both rules are almost the same except their reference to different objects in the knowledge base. The number of rules of a knowledge–based system depends on the number of objects and increases proportionally with the increase in the number of objects. To reduce the number of rules, all those rules which are similar in construction are combined into a generic rule using variables. An example of combining the two rules in Section 4.3.1 into a single rule with variables is shown below:

(MACHINE.ASSIGNMENT.RULE

(IF (the CURRENT.PROCESS of PART.A is ?CURRENT.PROCESS)
(the PROCESS.CAPABLE of ?MACHINE is ?PROCESS.CAPABLE)
(equal ?CURRENT.PROCESS ?PROCESS.CAPABLE)
THEN
(the MACHINE.TO.SEND.TO of PART.A is ?MACHINE)))

The variables used in the rule are preceded by a question mark such as ?CURRENT.OPERATION, ?MACHINE, and ?PROCESS.CAPABLE Specific values are bound to these variables when the rule is fired; for example, the value

in the CURRENT.PROCESS slot of PART.A object is bound to the variable ?CURRENT.PROCESS. All the objects in the knowledge base which have the PROCESS.CAPABLE slot are assigned to the variable ?MACHINE one at a time and the value in this slot is assigned to the variable ?PROCESS.CAPABLE. If all the premises are true for a certain object which is bound to the variable ?MACHINE, then the conclusion puts this object's name into the MACHINE.TO.SEND.TO slot of PART.A object. Thus the number of rules can be reduced by grouping similar rules into a single generic rule using variables thereby reducing the size of the rule base. When new objects are added to the knowledge base, there is no need to write new rules to reason about these objects. New rules are written only when the rule base is to be updated.

4.3.1.2 Rule classes and rule units

In KEE, rules are always organized into rule classes as follows:

(i) a single rule class which contains all the rules needed for solving the problem, or

(ii) a set of rule classes where each rule class contains a specific set of rules to solve a particular sub−problem.

By organizing the rules into several rule classes, only those rules belonging to a specific rule class have to be fired to solve a sub−problem thereby reducing the solution search space. For example, consider the rules for designing manufacturing systems. Instead of having a single rule class containing all the rules, we could have more than one rule class each containing a set of rules to solve a problem: a rule class to select machines, a rule class to select material handling systems, and so forth. The rule classes and rules (also called rule units in KEE) for this example are shown in Figure 4.12.

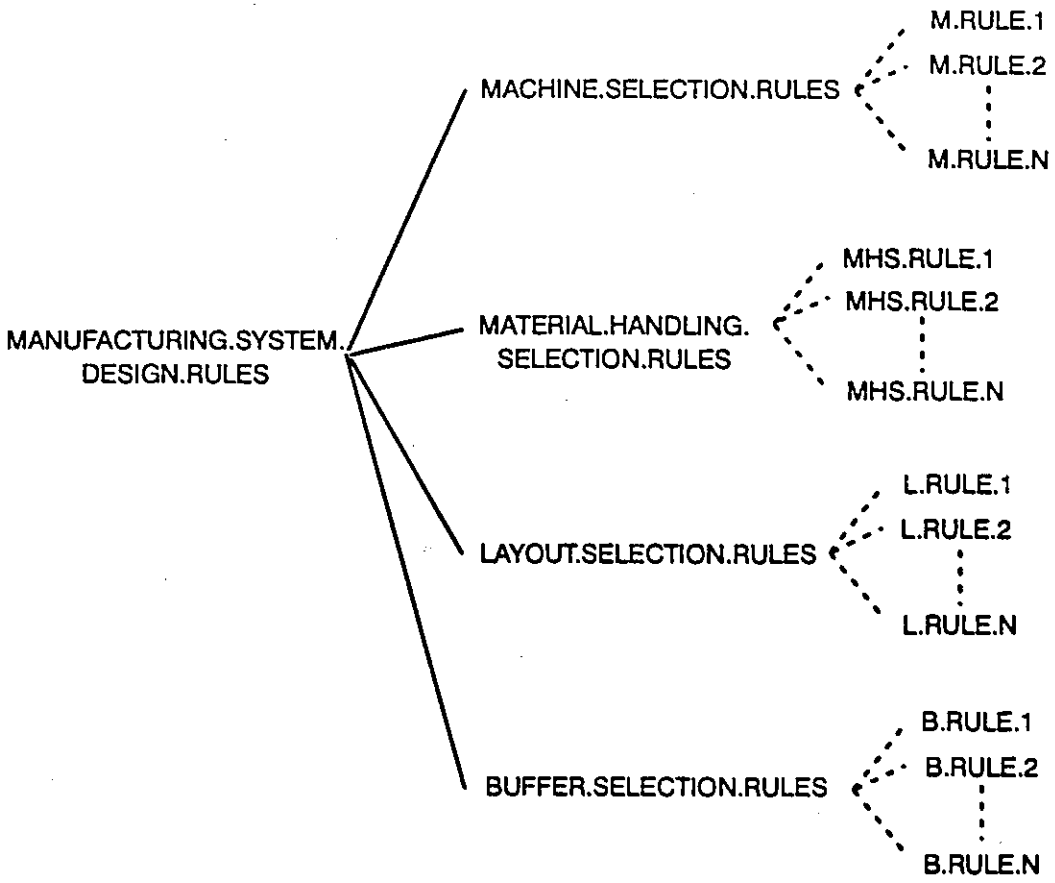


Figure 4.12. Example of an hierarchy of rule classes and rule units

The rule units are created as the children of the rule classes, for example, M.RULE.1, M.RULE.2, ..., M.RULE.N are rules units for the rule class MACHINE.SELECTION.RULES; MHS.RULE.1, MHS.RULE.2, ..., MHS.RULE.N are rule units for the rule class MATERIAL.HANDLING.SELECTION.RULES; and so forth.

4.3.2 Methods in KEE

At the heart of object–oriented programs are objects which have slots

describing their properties and behaviors. Properties define all the data related to the object while behaviors define how an object behaves under different conditions. Object slots containing procedural behaviors are called method slots and these are activated by sending a message to the slot.

A method is essentially a piece of computer code written in LISP and is attached to the slot of an object. There are two ways of providing behavior to an object using a method:

(i) The first approach (Figure 4.13.a) is to write the code in LISP and attach it to the slot of a high−level object. This code is then automatically inherited by all objects which are subclasses and members of the high−level object. In some situations methods can be wrapped with additional code around the basic code to modify the behavior of some objects. The wrapper code can then be called either before or after the original method code's procedures.

(ii) The second approach is to write the code in LISP and give it a name (usually a function name). This name is then assigned to the value of a method slot so that whenever a message is sent to this slot the function is executed. Methods that are created using this approach are easier to maintain and debug. Moreover, different objects can have the same method by having the function name of this method as a value of their method slot as shown in Figure 4.13.b.

4.3.3 Design synthesizer module of FMX

The main rule class in the design synthesizer module of FMX is FMS.DESIGN.SYNTHESIS.RULES. This rule class consists of four main rule classes:

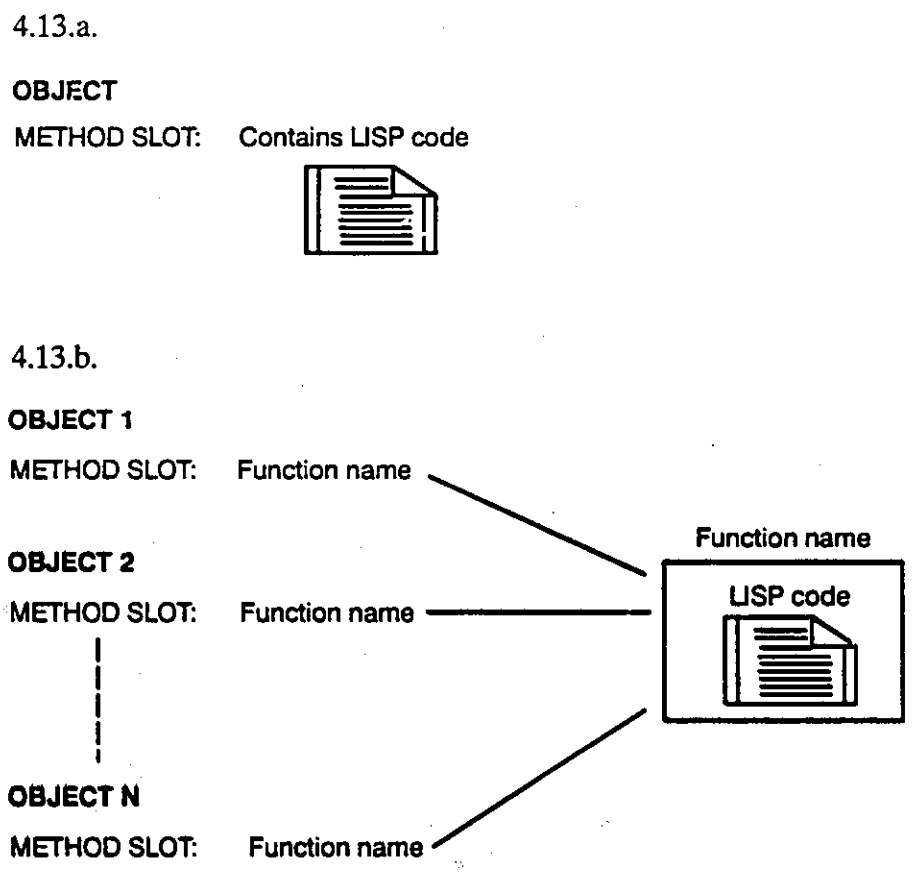


Figure 4.13. Types of method slots

(i) BUFFER.SELECTION.RULES,

(ii) MHS.AND.LAYOUT.SELECTION.RULES,

(iii) PALLET.SELECTION.RULES, and

(iv) WORKSTATION.SELECTION.RULES.

These rule classes and their hierarchy are illustrated in Figure 4.14. All the rules in the PALLET.SELECTION.RULES class are fired first, followed by the rules in the WORKSTATION.SELECTION.RULES class, then the rules in the MHS.AND.LAYOUT.SELECTION.RULES class, and finally by the rules

BUFFER.SELECTION.RULES class. Each rule class mentioned earlier has rule classes and rule units. A forward chaining strategy was adopted while firing the rules in each rule class.

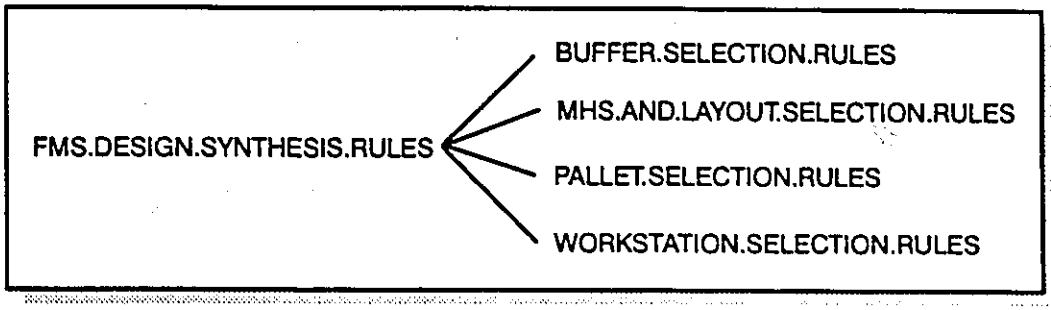


Figure 4.14. Main rule classes in the design synthesizer

4.3.3.1 Rules for selecting pallets

The rules of the rule class PALLET.SELECTION.RULES are shown in Figure 4.15. This rule class is not divided into any subclasses. The rules belonging

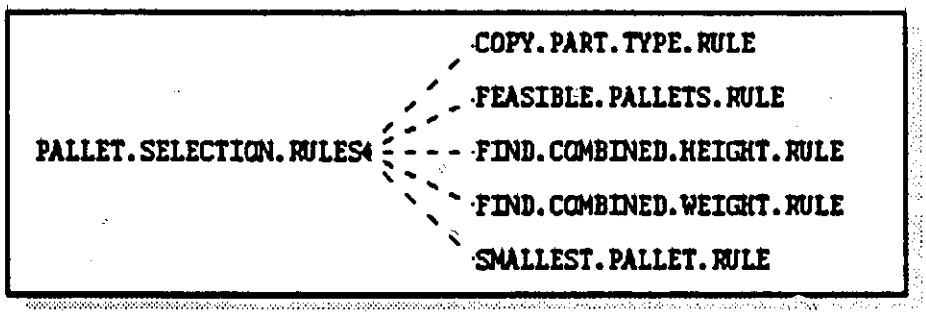


Figure 4.15. PALLET.SELECTION.RULES rule class

to this rule class select a pallet type for carrying each part type in the family. This is done by initially selecting a feasible set of pallet types for each part type and then

selecting the smallest pallet type in this feasible set. For example, the FEASIBLE.PALLETS.RULE selects a feasible set of pallet types for each part type in the family. The TellAndAsk syntax of this rule is shown below:

```
(FEASIBLE.PALLETS.RULE
   (IF   (ALL ?PART.TYPE ARE PART.TYPES)
         (THE LENGTH OF ALL ?PART.TYPE IS ?LENGTH)
         (THE BREADTH OF ALL ?PART.TYPE IS ?BREADTH)
         (ALL ?PALLET ARE PALLETS)
         (THE PALLET.DIMENSIONS OF ALL ?PALLET IS
               (LIST.OF (?PALLET.LENGTH ?PALLET.BREADTH)))
         (LISP (<= ?LENGTH ?PALLET.LENGTH))
         (LISP (<= ?BREADTH ?PALLET.BREADTH))
         THEN
         (THE PALLET.TYPE OF ALL ?PART.TYPE IS ?PALLET)))
```

4.3.3.2 Rules for selecting workstations

The rule classes and rules of the rule class WORKSTATION.SELECTION.RULES are shown in Figure 4.16. This rule class is divided into subclasses where each rule class contains a set of rules to perform an activity. The rules belonging to this rule class select the workstation types, determine their quantities, and generate the routes for each part type. For example, rules in the FEASIBLE.WORKSTATIONS.SET.RULES class select the feasible workstation types to process the family of part types. The rule PROCESS.CAPABILITY.RULE belonging to this rule class selects a set of

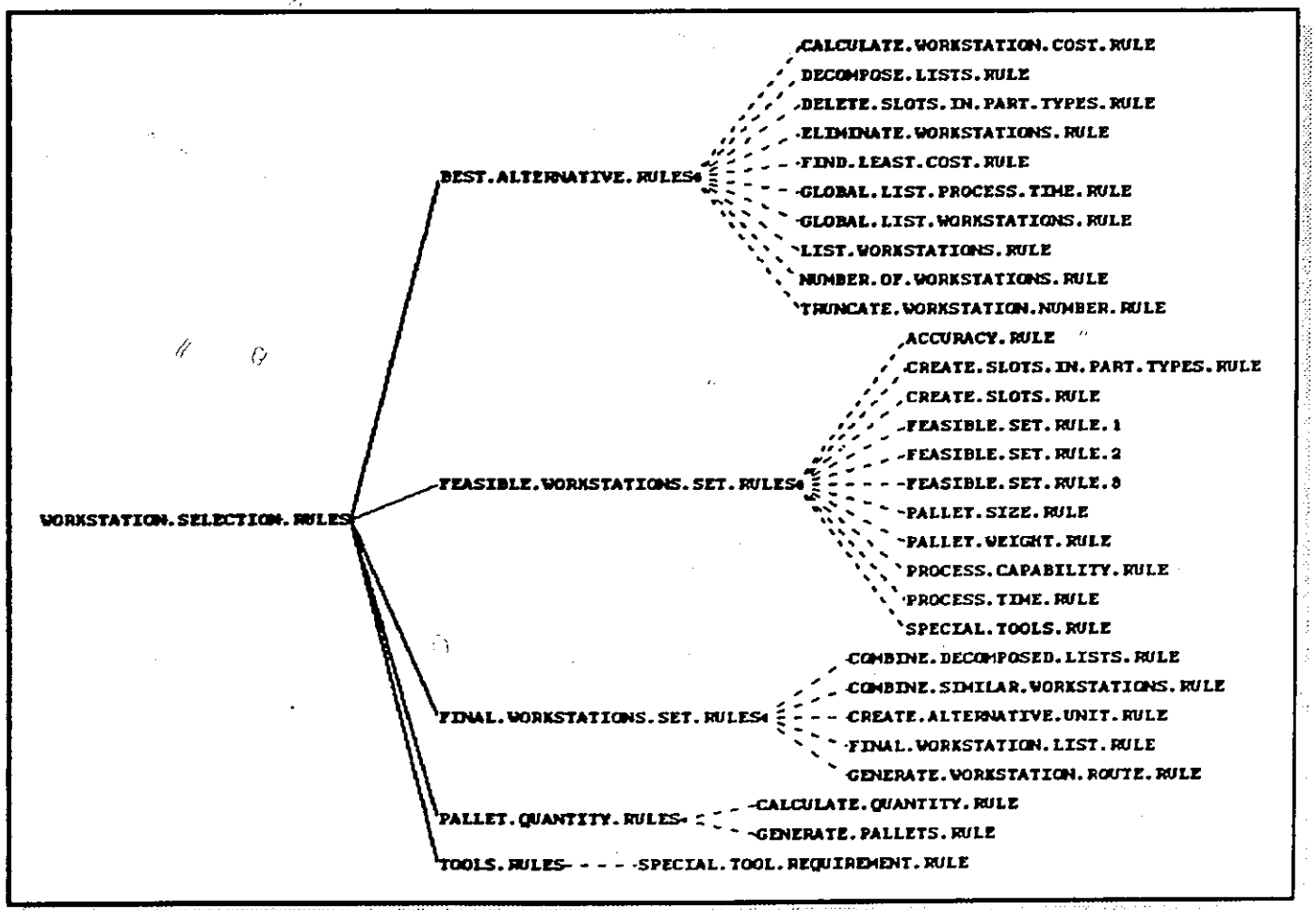


Figure 4.16. WORKSTATION.SELECTION.RULES rule class

workstation types that are capable of performing a process in the process sequence of each part type. The TellAndAsk syntax of this rule is shown below:

```
(PROCESS.CAPABILITY.RULE
    (IF    (ALL ?PART.TYPE ARE PART.TYPES)
           (THE PROCESS.SEQUENCE OF ALL ?PART.TYPE IS ?PROCESS)
           (THE SET.OF.POSSIBLE.WORKSTATIONS OF ALL ?PART.TYPE
                IS ?WORKSTATION)
           (THE PROCESS.CAPABILITY OF ALL ?WORKSTATION IS
                ?PROCESS.CAPABILITY)
           (LISP (EQUAL ?PROCESS ?PROCESS.CAPABILITY))
           THEN
           (LISP (UNITMSG 'SYNTHESIZER.CONTROLLER
                'CREATE.PROCESS.SLOTS! ?PART.TYPE ?PROCESS))
           (THE ?PROCESS OF ALL ?PART.TYPE IS ?WORKSTATION)))
```

On the other hand, rules in the FINAL.WORKSTATIONS.SET.RULES class generate the final list of workstations to process the family and establish the routing for each part type. The rule GENERATE.WORKSTATION.ROUTE.RULE calls a method that generates the workstation routing for each part type. The TellAndAsk syntax of this rule is shown below:

```
(GENERATE.WORKSTATION.ROUTE.RULE
    (IF    (ALL ?PART.TYPE ARE PART.TYPES)
           THEN
```

(LISP (UNITMSG 'SYNTHESIZER.CONTROLLER
'GENERATE.WORKSTATION.ROUTES! ?PART.TYPE))))

4.3.3.3 Rules for selecting material handling systems and layout

The rule classes and rules of the rule class MHS.AND.LAYOUT.SELECTION.RULES are shown in Figure 4.17. This rule class is further divided into subclasses where each rule class contains a set of rules to perform an activity. The rules belonging to this rule class select the appropriate material handling and system layout for the application under consideration. For example, rules in the rule class MHS.TYPE.RULES select the type of material handling system. The rule AGV.RULE selects AGVs as the type of material handling system based on the desired system flexibilities. The TellAndAsk syntax of this rule is shown below:

(AGV.RULE
(IF (THE ROUTING.FLEXIBILITY OF SYSTEM.DETAILS IS HIGH)
(THE VOLUME.FLEXIBILITY OF SYSTEM.DETAILS IS HIGH)
(THE EXPANSION.FLEXIBILITY OF SYSTEM.DETAILS IS HIGH)
THEN
(THE TYPE.OF.MHS OF SYNTHESIZER.MHS.CONTROLLER IS AGVS)))

Rules in the rule class LAYOUT.TYPE.RULES select the type of layout, line or loop, for the flexible manufacturing system. The rule LINE.RULE.1 selects a line

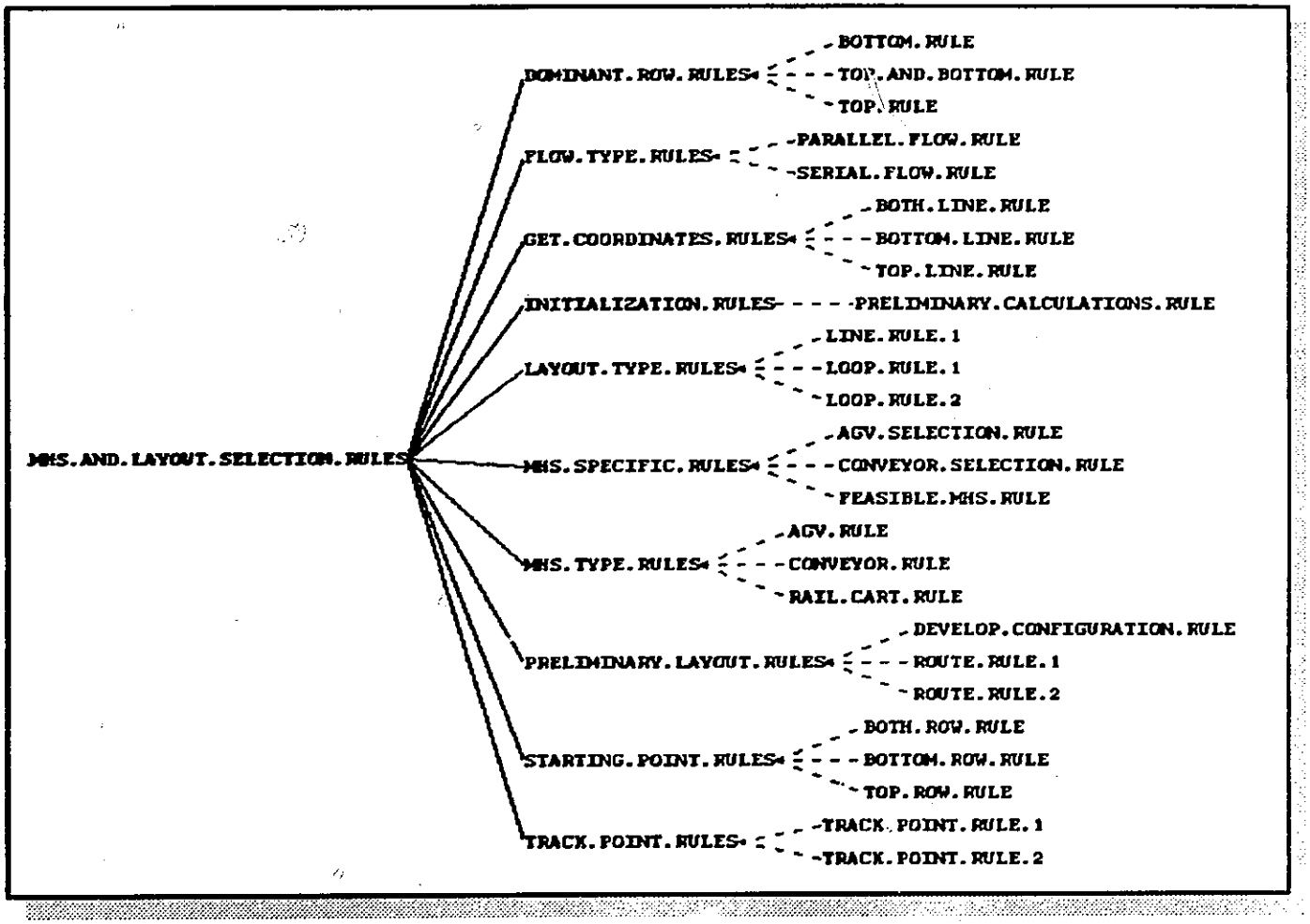


Figure 4.17. MHS.AND.LAYOUT.SELECTION.RULES rule class

layout based on the type of work flow and the material handling system. The TellAndAsk syntax of this rule is shown below:

```
(LINE.RULE.1
   (IF    (THE TYPE.OF.FLOW OF SYNTHESIZER.MHS.CONTROLLER
               IS PARALLEL)
          (FIND (THE TYPE.OF.MHS OF
               SYNTHESIZER.MHS.CONTROLLER IS ?MHS.TYPE)
               USING MHS.TYPE.RULES)
          (EQUAL ?MHS.TYPE RAIL.CARTS)
          THEN
          (THE TYPE.OF.LAYOUT OF SYNTHESIZER.MHS.CONTROLLER
               IS LINE)))
```

4.3.3.4 Rules for selecting automatic pallet changers

The rule classes and rules of the rule class BUFFER.SELECTION.RULES are shown in Figure 4.18. This rule class is further divided into two subclasses. The rules belonging to this rule class select the automatic pallet changer for each workstation. For example, the rule class APC.CAPACITY.RULES determine the capacity of the automatic pallet changer at each workstation. The rule APC.CAPACITY.RULE.1 determines the minimum capacity of an automatic pallet changer at each workstation based on the total number of pallets circulating in the system. The TellAndAsk syntax of this rule is shown below:

```
(APC.CAPACITY.RULE.1
```

(IF (THE TOTAL.NUMBER.OF.WORKSTATIONS OF
SYNTHESIZER.MHS.CONTROLLER IS ?WKS)
(THE TOTAL.NUMBER.OF.PALLETS OF
SYNTHESIZER.MHS.CONTROLLER IS ?PALLETS)
(LISP (> (FLOAT (/ ?PALLETS ?WKS)) 0.0))
(LISP (<= (FLOAT (/ ?PALLETS ?WKS)) 1.0))
THEN
(THE APC.CAPACITY OF SYNTHESIZER.MHS.CONTROLLER
IS 2)))

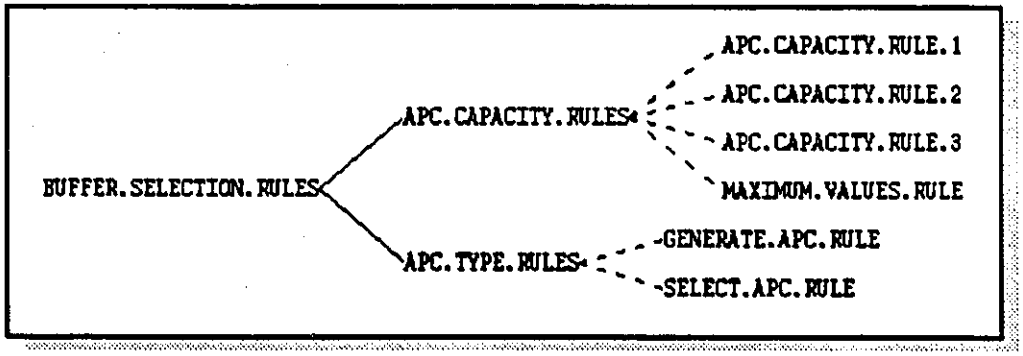


Figure 4.18. BUFFER.SELECTION.RULES rule class

4.4 Discussion

Building an expert system comprises of a process for obtaining knowledge (knowledge acquisition) and then implementing the means for utilizing this knowledge. Knowledge acquisition takes a significant amount of time and hence is the bottleneck process in the building of an expert system. A recent survey of knowledge engineers from private organizations confirmed this fact [52].

Knowledge acquisition involves the transfer and transformation of problem solving expertise from knowledge sources to computer programs. The quality of an expert system and its development cost depend primarily on the validity and reliability of techniques to acquire knowledge from various sources.

Potential knowledge sources include empirical data, case studies, text books, databases, human experts, and one's own experience. Traditionally, the following three techniques have been used to acquire knowledge from experts: unstructured interviews [53], protocol analysis [54], and Kelly's Repertory Grid [55]. These techniques, however, lack the theoretical support to comprehend and interpret an expert's verbal data. Abdul−Gader and Kozar [56] proposed a coherence method based on artificial intelligence planning theory and coherence theory to augment protocol analysis and interviews. They tried to provide a good theoretical basis for knowledge acquisition and proved that the coherence method facilitates the understanding of an expert's discourse in the early stages of the interview. The interview technique is the most widely used for knowledge acquisition and Agarwal and Tanniru [57] compared unstructured interviews to structured interviews. Results showed that the structured interview was better than the unstructured one.

Although significant improvements have been made in the developments of techniques for acquiring knowledge from experts, knowledge acquisition is still and error prone and time−consuming process. In this research, a few select experts were interviewed using a structured interview format. However, the majority of knowledge for various modules of FMX was acquired from textbooks, reported case studies, and author's own experience.

The input to the design synthesizer is the information about the part types and system requirements. This information is initially stored in a different

knowledge base before synthesis. Once design synthesis starts, this information is copied into the design synthesizer module and is used by the rules and methods of the design synthesizer to select the appropriate components from the component library, determine their quantity, and identify the type of layout for combining these components together to form the system.

The output from the design synthesizer is a list of selected components (pallets, machines, material handling systems, etc.), the relationships between these components, and the approximate positions of these components in the layout. At this point in the FMS design process, a graphical model does not exist. It is developed by the simulation model developer which is explained in the next chapter.

The problem solving ability of an expert system is a function of the number of rules that are used to develop the system. In a production system it is easy to determine the number of rules because each rule is unique and there is no generic rule that can replace a class of rules. The lack of an object hierarchy in such systems prevents the use of generic rules which can reason about a group of objects. The addition of an object always necessitates the addition of unique rules to reason about this object.

On the other hand, it is very difficult to exactly determine the number of rules in an object-oriented expert system. This is largely due to the use of generic rules to reason about one or more classes of objects. As mentioned in Section 4.3.1.1, generic rules are obtained when rules which are similar in construction are combined by using variables. The addition of new objects to the hierarchy does not necessitate the change of rules or addition of new rules.

The design synthesizer module of FMX currently has approximately 70 generic rules which use a forward chaining approach to synthesize new flexible manufacturing systems. Although these rules do not cover the entire spectrum of

heuristics used for synthesizing new designs, they consider many important features of flexible manufacturing system components for generating initial designs. Moreover, the rules are used for designing flexible manufacturing systems in the machining environment. The rule base of the design synthesizer could be expanded to improve its performance and to extend the scope of FMX to other manufacturing areas such as assembly or metal forming. However, the addition of new rules to an object–oriented expert system like the design synthesizer is a complex process. Rules which are in English–like format have to be converted to a format that the software recognizes, in this case the TellAndAsk syntax. This requires a thorough knowledge of the software used to develop the expert system, the object–oriented structure of objects and rules of the existing system, and the inference process. However, this is done by the software system developer and not the flexible manufacturing system designer.

CHAPTER 5

MODEL DEVELOPER AND SIMULATOR

This chapter describes the two major activities of discrete–event simulation, model development and model execution, in detail. It proceeds to highlight the drawbacks of the traditional approach to develop simulation models using simulation languages and an alternate automated approach is presented. The various principles involved in the development of a simulation model of an FMS design are then explained in detail. The implementation of the simulation model developer module of FMX using KEE and SimKit is presented. This chapter concludes with an overview of the model execution activity and the implementation of the simulator module of FMX.

5.1 Evaluation of FMS performance

The output from the design synthesizer is a list of components which represents the initial FMS design. Several model–based analysis tools such as analytical models, physical models, and discrete–event simulation models are available to evaluate the performance of this design. In this research, the performance of the initial FMS design as well as subsequent designs are evaluated using discrete–event simulation which consists of two major activities:

(i) model development which involves the construction of a model of the

flexible manufacturing system, and

(ii) model execution which involves running the model and generating output data.

5.2 Model development

Models are often used to study the behavior of complex systems because the system is conceptual and does not exist physically yet, or it is difficult and costly to experiment with the real system. One of the most widely used modeling approaches, especially for analyzing the behavior and evaluating the performance of manufacturing systems, is discrete–event simulation. It involves developing a model of the real system and experimenting with this model to evaluate the system under different experimental conditions.

Simulation models of real systems are developed by representing the system components and their relationships in an appropriate form. Two types of modeling representation are available:

(i) ***Iconic models:*** In these models, the components of the real system are represented as scaled–down physical models and the relationships between the various components are established by physically connecting them. The physical models mentioned in Section 2.3.2.2 are examples of iconic models.

(ii) ***Symbolic models:*** In these models, the components of the real system are represented abstractly in a computer. The relationships between the components are established by logical connections using a programming language rather than physical connections. Symbolic models are computer programs which can be modified easily and hence, are more flexible than

iconic models. The symbolic modeling approach was chosen for this research.

5.2.1 Simulation languages

A programming language is necessary to develop the symbolic model of a manufacturing system in a computer. The historical development of programming languages, especially for simulation, can be divided into five major periods [58]. Until 1960, all simulation models were written in high-level languages, such as FORTRAN. Many simulation models are still being written in other high-level languages, such as BASIC, PASCAL, or C. The second period, 1960-65, saw the birth of the first generation of simulation languages—languages specifically meant for simulating complex systems. These specialized simulation languages were preferred to high-level languages because they:

(i) reduced the programming effort,

(ii) resulted in fewer errors,

(iii) were easier to debug, and

(iv) facilitated automatic generation and collection of statistics.

The third period, 1966-70, saw enhancements to the simulation languages introduced in the second period. These enhancements made the languages more powerful and robust. The focus of the fourth period, 1971-80, was the development of languages which simplified the programming process. This was achieved by separating the model development, execution, and output analysis activities, and by using advanced data base management and computer graphics capabilities.

Currently, we are in the fifth period of simulation language development. The emphasis on this period is the application of artificial intelligence, especially expert system techniques, to the simulation modeling process. The focus is to

provide an integrated, intelligent simulation environment that combines expert systems and simulation rather than the development of powerful simulation languages.

5.2.2 Creating simulation models

The increased use of simulation as a tool for analyzing manufacturing systems can be largely attributed to the vast developments made in the areas of computer hardware and software during the fourth period, 1971–80. There are two approaches to the development of simulation models for manufacturing systems:

(i) the traditional approach and

(ii) the automated approach.

Computer simulation languages greatly facilitate the development and execution of these simulation models. Some of the commercial simulation languages that are used at present for modeling and analyzing manufacturing systems include SIMAN and SLAM II.

5.2.2.1 Traditional approach

The traditional approach to the development of a simulation model is illustrated in Figure 5.1. System designers study the system to be simulated and obtain data about it. They then establish a detailed specification of the system and use a simulation language, such as SIMAN or SLAM II, to develop a model of the system. The acquisition and programming tasks performed by the system designers are manual and they require knowledge about the system and the simulation language. Development of simulation models using the traditional approach, therefore, depends on the language used and requires system designers to be proficient in the modeling constructs of these languages.

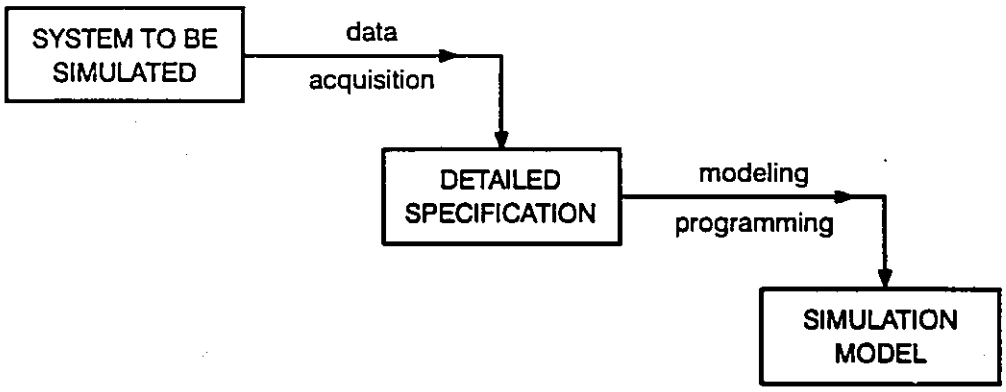


Figure 5.1. Traditional approach to simulation model development [59]

For example, consider a manufacturing system with one workstation as shown in Figure 5.2. Parts enter the system, wait their turn to be processed, and then

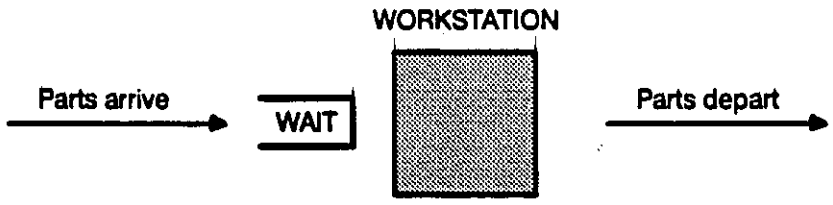


Figure 5.2. Schematic of a simple manufacturing system

leave the system. These parts enter the system one at a time with an exponentially distributed interarrival time with a mean of 4.4 minutes. The combined setup and machining time for each part on the workstation follows a triangular distribution with a minimum of 3.2 minutes, a mode of 4.2 minutes, and a maximum of 5.2 minutes. The system operates on a single, eight–hour shift during weekdays and shuts down over the weekend. The simulation model of this system to analyze its

performance was developed using SIMAN [60]. The model and experimental files that use SIMAN's modeling constructs are listed below:

Model file in SIMAN for sample problem

```
BEGIN;
    CREATE:     EXPONENTIAL(4.4);          Enter the system
    QUEUE,      Buffer;                    Wait for the workstation
    SEIZE:      Workstation;               Seize the workstation
    DELAY:      TRIANGULAR(3.2,4.2,5.2);   Delay for workstation
    RELEASE:    Workstation;               Release the workstation
    COUNT:      JobsDone:DISPOSE;          Count completed parts
END;
```

Experimental file in SIMAN for sample problem

```
BEGIN;
PROJECT,      Sample Problem, SM;
DISCRETE,     100;
QUEUES:       Buffer;
RESOURCES:    Workstation;
COUNTERS:     JobsDone;
REPLICATE,    1,0,480;
END;
```

System designers should possess adequate knowledge about the modeling constructs of SIMAN to develop these model and experimental files shown above. Acquiring this knowledge takes a considerable amount of time and effort.

5.2.2.2 Automated approach

Using the traditional approach, system designers spend a considerable amount of time to learn a specific language before developing simulation models. An approach which automates the acquisition and programming tasks has been

proposed to overcome this drawback of the traditional approach [59]. This approach, illustrated in Figure 5.3, consists of the following two steps:

(i) obtaining specific information about the system through a natural language (NL) processor, graphics interface, or dialog monitor, and

(ii) developing a simulation model automatically using an automatic programming system.

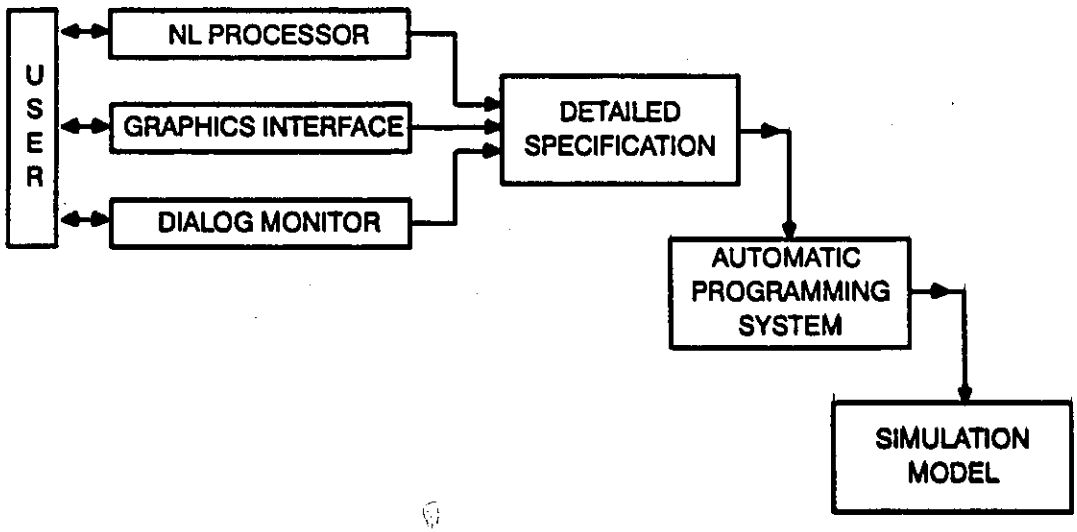


Figure 5.3. Automated approach to simulation model development [59]

The natural language processor poses a series of questions to the user in an English–like form. The response of the user provides the description of the system to be simulated. Although a major advantage of the natural language interface is its flexibility, its implementation, however, requires a great deal of linguistics knowledge. Moreover, constantly responding to a series of questions makes its use a tedious process.

The graphics interface, on other hand, permits a detailed description of a system by laying out the components that comprise the real system. A prerequisite

for this is the libraries of graphical icons, where each icon represents a component in the real system. Developing simulation models from using these libraries would then involve creating copies of these icons and establishing the relationships between them. A major drawback of the graphics interface is the need for an additional interface to provide details about the system such as distribution parameters, model termination conditions, and queue capacity. Therefore, natural language processor or a dialog monitor is often used in conjunction with the graphics interface for specifying the problem and developing the simulation model.

The automatic programming system is an expert system that consists of three distinct types of knowledge: domain knowledge, i.e., knowledge about queueing systems; general simulation modeling knowledge; and target language knowledge, in this case, SIMAN. These are stored in the form of condition–action rules, IF–THEN, and use the input provided by the user to generate the model and experimental files in SIMAN. The basic objective of the automated approach is to help system designers with minimum knowledge of simulation techniques and programming language expertise to build executable simulation models. System designers, therefore, can concentrate on the system being modeled rather than the mechanics of the model development process.

5.3 FMS simulation model development

Knowledge–based simulation systems are simulation languages that combine expert system techniques and simulation methodology. The object–oriented programming paradigm of these systems facilitates the development of a library of system components and the addition of properties and behaviors to these components by means of slots. The graphics facility aids the

development of icons for each component in the library. Therefore, a prerequisite for developing simulation models of flexible manufacturing systems is the development of a library of FMS components. This requires knowledge of the following:

(i) temporary and permanent entities,

(ii) relationships between permanent entities,

(ii) flow of temporary entities and their interaction with permanent entities, and

(iv) properties and behaviors of entities.

5.3.1 Temporary and permanent entities

From the viewpoint of simulation, a system is a collection of entities, both temporary and permanent. Temporary entities are those that are created at various points in time during the simulation and are destroyed later while permanent entities are those that remain constant throughout the entire simulation. These different types of entities have to be identified before developing a simulation model.

The major components of flexible manufacturing systems are pallets, automatic pallet changers, workstations, material handling systems, and part types. Part types are classified as temporary entities while the rest of the components are classified as permanent entities. Of the permanent entities, pallets and material handling systems are dynamic, i.e., they move from one position in the layout to another. Workstations and automatic pallet changers, on the other hand, are static, i.e., they remain stationary at specific positions in the layout.

5.3.2 Relationships between permanent entities

Before developing a simulation model of an FMS design, it is important to know about the physical connections of the various components, for example, which system components are connected to workstations, or which system components are connected to automatic pallet changers, etc. These connections establish the relationships and hence, the interactions between the various components.

Parts mounted on pallets move from the automatic pallet changer to the workstation for processing and vice versa after processing. Therefore, there is a two–way connection between a workstation and an automatic pallet changer. Similarly, loaded or empty pallets move from the automatic pallet changers to the material handling system or vice versa. Each automatic pallet changer is connected to a track point—a location on the shop floor in front of the automatic pallet changer [61]. Each track point is always connected to two other track points, either by conveyor segments or tracks, depending on the type of material handling system (Figure 5.4). For example, if conveyors are used to move the parts and pallets, then

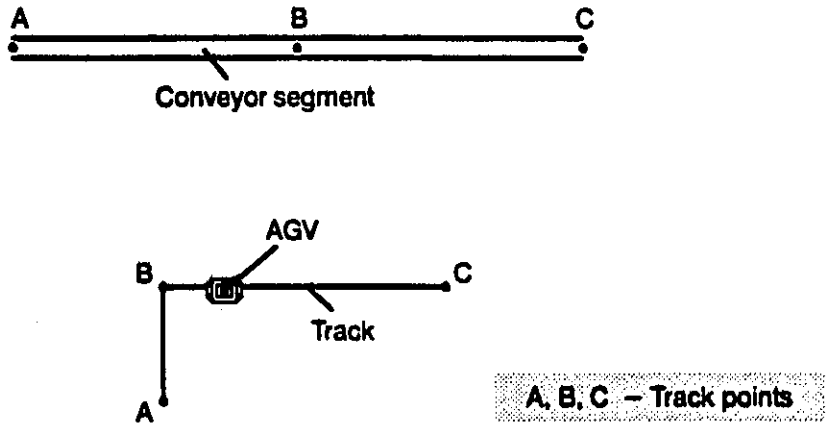


Figure 5.4. Connection between adjacent track points

the conveyor segments connect two adjacent track points. However, if AGVs or rail carts are used to transport the pallets, then tracks are used to connect two adjacent track points. The AGVs or rail carts travel on these tracks between two track points.

Loaded or empty pallets are physically connected to either the workstations, automatic pallet changers, or material handling systems. These connections are temporary and the component to which a pallet is connected to at a certain point in time depends on the state of the system at that time. These physical connections between the components of the FMS design have to be established before generating a graphical simulation model. Figure 5.5 is a schematic diagram which shows the connections between the various FMS components.

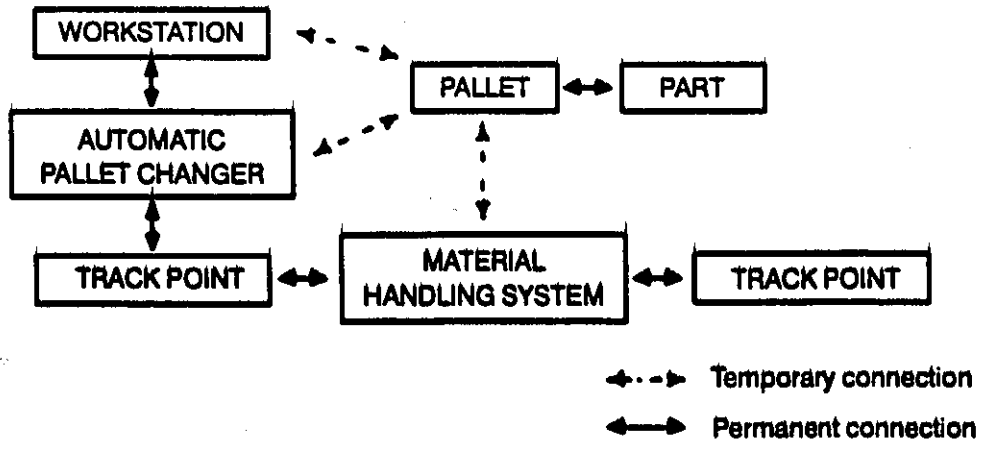


Figure 5.5. Physical connections between various FMS components

5.3.3 Part flow in a flexible manufacturing system

Each part (temporary entity) that flows through a flexible manufacturing system moves from one workstation to another depending on its routing. As the part

flows through the system, it interacts with other components constantly and changes the state of the system periodically. For example, a part increases the total number of parts in an automatic pallet changer upon its arrival and decreases the total number upon its departure. Similarly, a part changes the state of a workstation to busy upon its arrival and to idle upon its departure.

The flow of a part through a flexible manufacturing system is shown in Figure 5.6. As soon as a part enters the system, the first workstation it has to visit is identified and the part is sent to the automatic pallet changer that is connected to this workstation. The first process is always a loading operation in which the part is loaded on a pallet. If the workstation is available the part is immediately moved to the workstation table, if not it waits in the automatic pallet changer. After the process is completed, the part is transferred back to the automatic pallet changer if space is available. If not, the part waits in the workstation table and as a result, blocks other parts waiting for this workstation. Once the part is in the automatic pallet changer it waits for the material handling system to take it to the next workstation in its routing. If this is the last process in the process sequence, then the part is removed from its pallet and it departs the system.

5.3.4 Properties and behaviors of entities

The object-oriented library, FMSLIB, which contains the major components of flexible manufacturing systems was explained in Section 3.3.2 and properties that were useful for design synthesis were mentioned. The development of simulation models of flexible manufacturing systems, however, requires that these components in FMSLIB be modeled as elements of a queueing network. By making FMSLIB a sublibrary of QLIB, as shown in Figure 3.16, the properties and behaviors of objects in QLIB are automatically inherited by its children in FMSLIB.

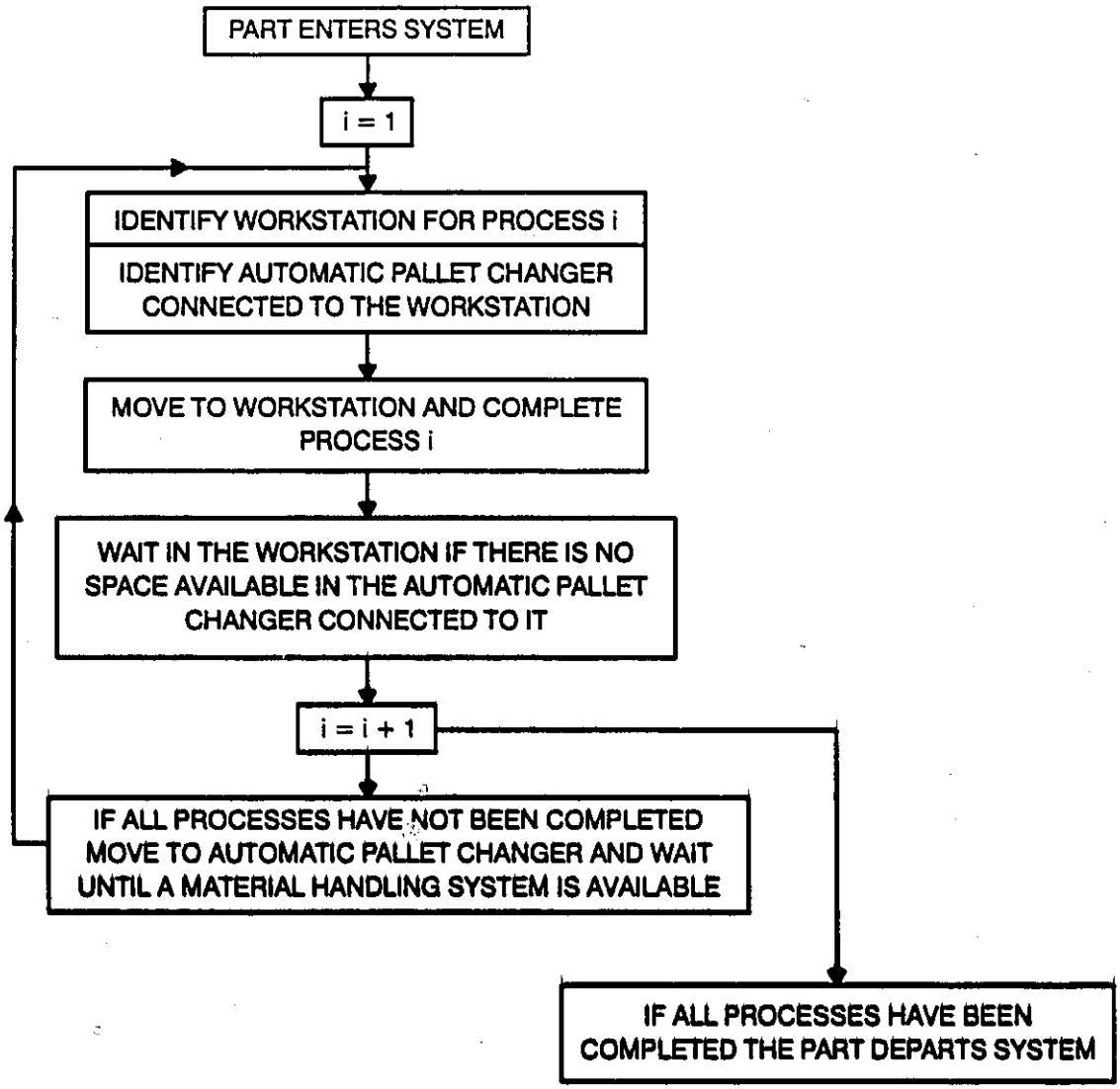


Figure 5.6. Part flow in a flexible manufacturing system

For example, the object NCVTL.A in FMSLIB which is a child of the SINGLER.SERVERS object in QLIB inherits both method and non−method slots as shown in Figure 5.7.

The objects in QLIB are provided with slots which contain the properties and behavior of queueing systems. This is valid, however, for a generic queueing

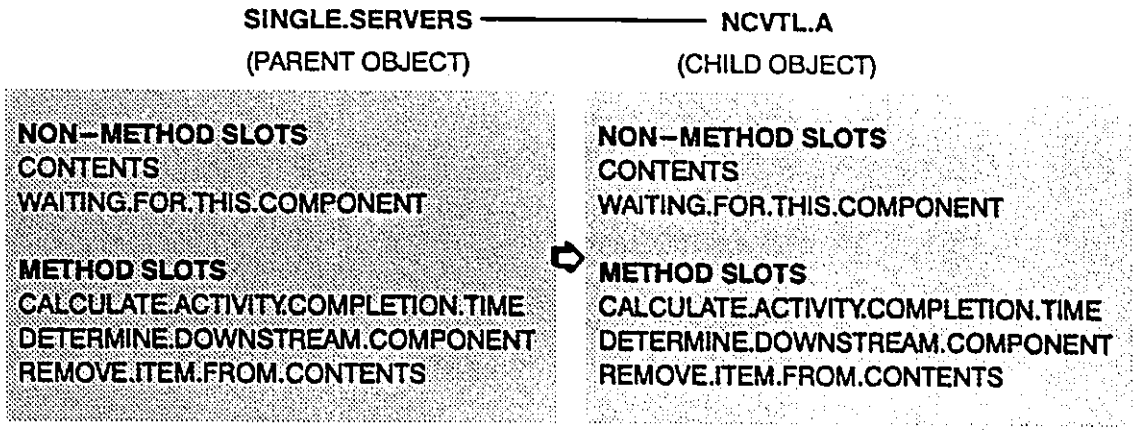


Figure 5.7. Inheritance of non–method and method slots

system and behaves in pre–defined ways. To apply it to a specific type of queueing system, such as a flexible manufacturing system, the slots of objects in FMSLIB should be modified. Three ways of modifying the FMSLIB library are listed below:

(i) changing the slot values of some non–method slots,

(ii) writing new functions and placing these as values of some existing method slots, or

(iii) making complex modifications such as creating new method and non–method slots, writing new functions, and creating new events.

5.3.4.1 Modifications by changing slot values

One way of making changes to FMSLIB is by changing the values of non–method slots. For example, the capacity of an automatic pallet changer is the number of parts it can hold at a time and is either 2, 4, or 6 depending on the type. Thus the capacity of the automatic pallet changer APC.2.A is 2 while the capacity of APC.4.A is 4 and that of APC.6.A is 6. All the automatic pallet changers in

FMSLIB are children of the FIFO.QUEUES (first–in first–out queues) object in QLIB and the value of the CAPACITY slot of FIFO.QUEUES is 10. By the property of inheritance, this value is inherited by the CAPACITY slots of all automatic pallet changers. Therefore, the value of the CAPACITY slot of each automatic pallet changer in FMSLIB was changed depending on its type as shown in Table 5.1.

Table 5.1. Original and new values of the CAPACITY slot

Automatic pallet changers	CAPACITY slot	
	Original value	New value
APC.2.A	10	2
APC.2.B	10	2
APC.2.C	10	2
APC.4.A	10	4
APC.4.B	10	4
APC.4.C	10	4
APC.6.A	10	6
APC.6.B	10	6
APC.6.C	10	6

5.3.4.2 Modifications to FMSLIB by writing new functions

Another way of making changes to FMSLIB is by writing new functions and placing the names of these functions as values of existing method slots. These new functions are identified by a name and are written in LISP. For example, consider the COMPLETE.ACTIVITY! method slot of the SINGLE.SERVERS object in QLIB. The value of this slot is COMPLETE.ACTIVITY! which is the name of a function written in LISP (Figure 5.8). The NCVTL.A object in FMSLIB is a subclass of the SINGLE.SERVERS object and automatically inherits the COMPLETE.ACTIVITY! slot and its value. However, the behavior when an

activity is completed at a workstation is different from that at a server. Therefore, a new function, COMPLETE.ACTIVITY.ON.SERVER!, was written in LISP and the name of this function was put as the value of the COMPLETE.ACTIVITY! slot of the NCVTL.A object (Figure 5.9).

```
(Output) The SINGLE.SERVERS Unit in QLIB Knowledge Base
Member slot: COMPLETE.ACTIVITY! from QLIB.COMPONENTS
   Inheritance: METHOD
   ValueClass: METHOD
   Comment: "Event that occurs when an activity is completed by//on a
            component"
   Values: COMPLETE.ACTIVITY!
```

Figure 5.8. COMPLETE.ACTIVITY! slot of the SINGLE.SERVERS object

```
(Output) The NCVTL.A Unit in FMSLIB Knowledge Base
Member slot: COMPLETE.ACTIVITY! from NCVTL.A
   Inheritance: METHOD
   ValueClass: METHOD
   Comment: "Event that occurs when an activity is completed by//on a
            component"
   Values: COMPLETE.ACTIVITY.ON.SERVER!
```

Figure 5.9. COMPLETE.ACTIVITY! slot of the NCVTL.A object

Similarly, the ITEM.DEPARTS! method slot of the FIFO.QUEUES object in QLIB has a value ITEM.DEPARTS.FROM.FIFO.QUEUES! as shown in Figure 5.10. The APC.6.A object in FMSLIB which is a subclass of the FIFO.QUEUES object, however, has a value ITEM.DEPARTS.FROM.APC! as shown in Figure 5.11.

```
(Output) The FIFO.QUEUES Unit in QLIB Knowledge Base
Member slot: ITEM.DEPARTS! from FIFO.QUEUES
   Inheritance: METHOD
   ValueClass: METHOD
   Comment: "Event that occurs when an item departs from a component"
   Values: ITEM.DEPARTS.FIFO.QUEUES!
```

Figure 5.10. ITEM.DEPARTS! slot of the FIFO.QUEUES object

```
(Output) The APC.6.A Unit in FMSLIB Knowledge Base
Member slot: ITEM.DEPARTS! from APC.6.A
   Inheritance: METHOD
   ValueClass: METHOD
   Comment: "Event that occurs when an item departs from a component"
   Values: ITEM.DEPARTS.FROM.APC!
```

Figure 5.11. ITEM.DEPARTS! slot of the APC.6.A object

5.3.4.3 Complex modifications

A third and a more complex way of making changes to FMSLIB is by creating new method or non−method slots, new functions, or new events. For example, the CENTROID.WRT.PICTURE slot of the AGV.A object is a new non−method slot which contains the coordinate position of the centroid of this object. The slot which is shown in Figure 5.12 has an active value attached to it which automatically calculates the coordinates of the object and moves it to a new location whenever a new centered position is entered. The "Cardinality.Max" facet of this slot constrains it to have a maximum of one value while the "Cardinality.Min" facet of this slot constrains it to have a minimum of one value. Thus this slot can either have only one value or no value at all.

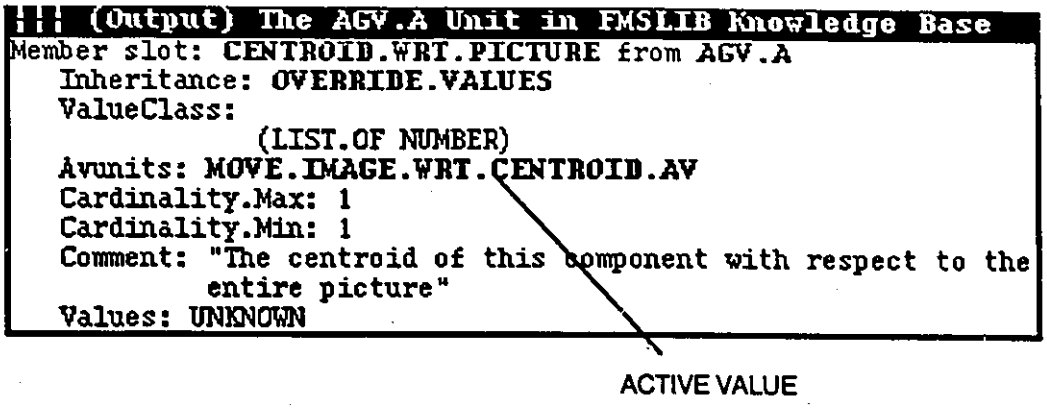


Figure 5.12. CENTROID.WRT.PICTURE slot of the AGV.A object

New method slots were created and new functions were written to model the behavior of material handling systems and pallets. For example, consider the ROTATE.MHS.BITMAP! method slot of the RAIL.CART.C object with a value ROTATE.MHS.BITMAP as shown in Figure 5.13. Whenever a message is sent to

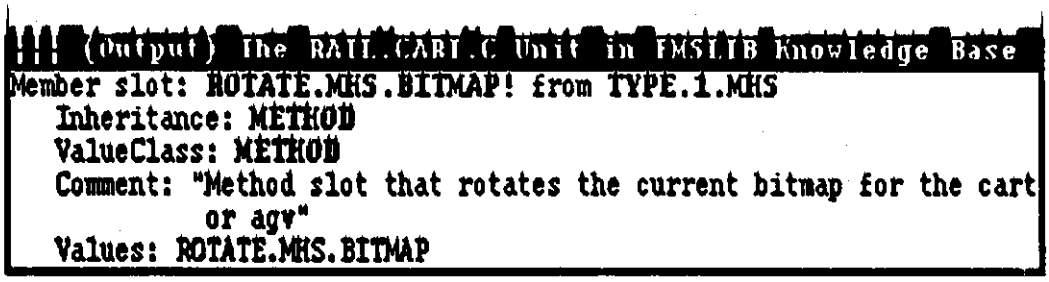


Figure 5.13. ROTATE.MHS.BITMAP! slot of the RAIL.CART.C object

this slot, the image of the object is rotated by 90 degrees. This new method slot is attached to AGVs and rail carts because the images of these objects have to be rotated by 90 degrees whenever they reach a corner track point in the layout.

New events were written to animate the movement of the pallets and the material handling systems in the graphical display. For example, the function

SCHEDULE.EMPTY.CART.MOVEMENT schedules an empty AGV or rail cart to move from its current position to a new one. It calculates the time taken to reach this new position and also schedules a new event, EMPTY.CART.DEPARTS!, which is the departure of the AGV or rail cart from the current position. This event in turn schedules another new event, EMPTY.CART.ARRIVES!, which is the arrival of the AGV or rail cart at its new position. This event is scheduled to take place at a time which is the sum of the current time and the time taken to reach the new position. Animation, in addition to providing a good understanding of the system, also helps in verifying the model.

5.3.5 Developing a graphical simulation model

Knowledge–based simulation systems use graphics interfaces extensively to build simulation models. A prerequisite for this is the development of an object–oriented library of flexible manufacturing system components with an icon attached to each component and slots to describe the properties and behavior of each component. The actual development of a graphical simulation model involves making copies of the appropriate icons in the library and establishing the relationships between these icons. The result is a graphical simulation model of the original system.

The development of a simulation model using this approach becomes very complicated when there are several components in the actual system. The graphical display of this model gets cluttered with icons and does not resemble the layout of the actual system. An example of a cluttered display is shown in Figure 5.14. Another drawback of this approach is the amount of time spent for developing graphical simulation models. One of the objectives of this research is to

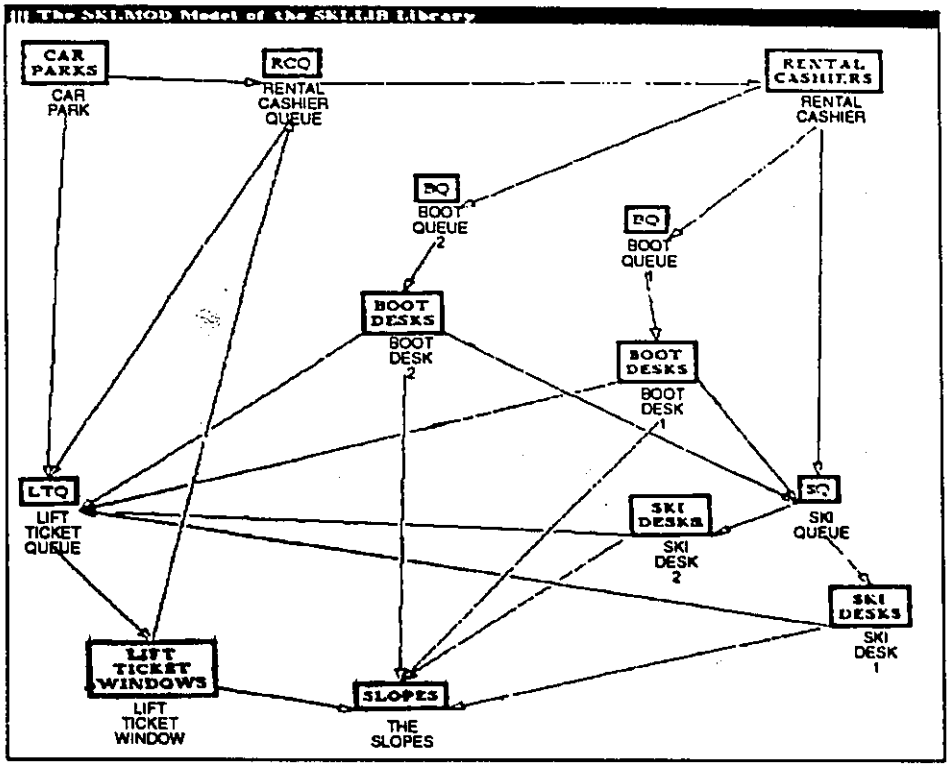


Figure 5.14. Example of a cluttered graphical simulation model [45]

automatically create a graphical simulation model of an FMS design in which the model is a graphical display that closely resembles the system.

To automatically generate the graphical simulation model of a design, the icons which represent the various system components have to be positioned at the appropriate coordinate positions in the layout. Each icon is provided with a reference point to ensure its correct placement and in this work the centroid of an icon is chosen as its reference point. The centroid of the NCMM.A object is shown in Figure 5.15.

Each workstation in the FMS design is connected to an automatic pallet changer and vice versa. Each automatic pallet changer is then connected to a track

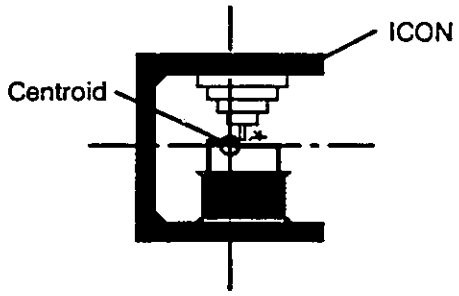


Figure 5.15. Centroid of the NCMM.A icon

point and vice versa. The coordinates of all the track points are determined first. The coordinates of all the automatic pallet changers which are connected to the track points are then determined. Finally, the coordinates of the workstations that are connected to the automatic pallet changers are determined. The relationships used for calculating the coordinates of the various system components is illustrated in Appendix C.

The automatic generation of the graphical model starts by displaying the track points followed by a display of the automatic pallet changers connected to these track points. The workstations which are connected to the automatic pallet changer are then visualized. The track points are connected to each other by conveyor segments or tracks depending on the type of material handling system. In the case of AGVs or rail carts, the tracks between track points are displayed followed by positioning each AGV or rail cart at a track point. In the case of conveyors, the conveyor segments between the track points are displayed. Finally, the pallets are positioned in the layout. While all the other icons are displayed in the layout, the pallets are not displayed initially. They are, however, displayed once the simulation of the model is started.

5.3.6 Data collectors

One of the most important events that takes place during a simulation study is the collection of data which is generated as the simulation progresses. This data is used to generate statistical reports and graphs on parameters of interest such as the average utilization of workstations and material handling systems, average number of parts waiting for the workstations, and average number of parts waiting for the material handling system. In knowledge–based simulation systems, this is done by attaching data collectors to slots of objects. Two types of data are collected during a simulation study:

(i) ***Numeric values:*** These are numbers that are collected by a slot during a simulation. For example, a slot which keeps track of the number of parts in an automatic pallet changer collects numeric values.

(ii) ***States:*** These are non–numeric and indicate the state of a particular slot. For example, a slot which keeps track of the state of workstation collects non–numeric values such as busy, idle, or failed.

The following types of data collectors are available in SimKit [45] to collect different types of data:

(i) STATE.FREQUENCY.DISTRIBUTION.COLLECTORS,

(ii) STATE.TIME.DISTRIBUTION.COLLECTORS,

(iii) INTERVAL.FREQUENCY.DISTRIBUTION.COLLECTORS,

(iv) INTERVAL.TIME.DISTRIBUTION.COLLECTORS,

(v) NUMERIC.COLLECTORS,

(vi) TRACE.COLLECTORS, and

(vii) WEIGHTED.MEAN.COLLECTORS

The type of data collector depends on the type of data collected by the slot

to which its is attached. For example, the data collector for the status of a workstation is different from the one to collect information about the average queue length of parts waiting for the workstation.

5.4 Implementation of the model developer

The object–oriented programming paradigm of KEE and SimKit was used extensively to automatically develop the simulation model and graphically display the model. The various functions, such as the creation of a window whose dimensions are proportional to the floor plan of the system, the creation and display of track points, automatic pallet changers, workstations, and material handling systems, were written in LISP and attached to method slots. These functions are activated by sending messages to these method slots. For example, the message to develop a graphical simulation model activates other messages such as the message to create a window, the message to create and display track points, message to create and display automatic pallet changers, message to create and display workstations, message to create and display material handling systems, and message to create and display pallets. This process of sending messages is illustrated in Figure 5.16. Sample displays of the simulation model development are shown in Figure 5.17.

After the graphical simulation model has been created and displayed, data collectors are attached to the slots of components to collect data. For example, to monitor the status of each workstation, a data collector is attached to the STATUS slot of each workstation in the model. Because the value of this slot is a state, either busy, idle, failed, or blocked, the data collector attached to the STATUS slot of each workstation is of the type STATE.TIME.DIST.COLLECTORS. This type of data

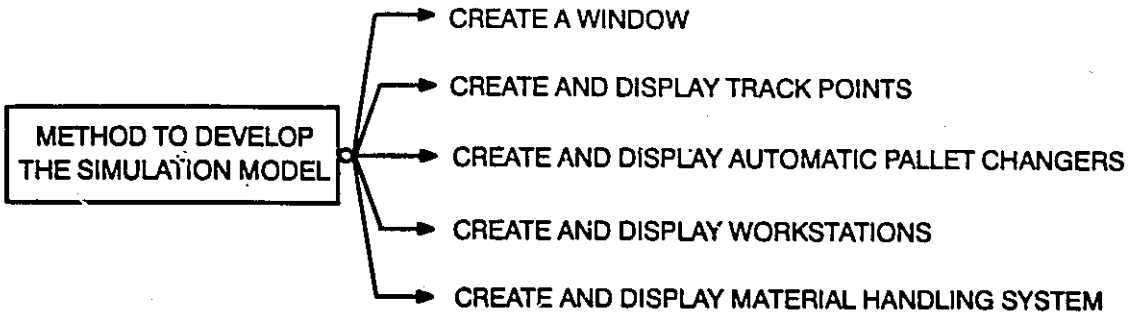


Figure 5.16. Messages to create the graphical simulation model

collector shows the length of simulated clock time for which a variable has a certain value. Figure 5.18 shows the STATUS slot of a workstation NCMM.A.2 before and after a data collector has been attached to this slot.

Consider the NUMBER.FOR.WKS slot of automatic pallet changer APC.2.B.1 which contains information about the number of parts waiting for a workstation. This slot stores only numbers. By attaching a suitable data collector to this slot, the average waiting time for a part is calculated at the end of the simulation. The data collector attached to the slot for this purpose is of the type WEIGHTED.MEAN.COLLECTORS which looks at numerical values in relation to time. Figure 5.19 shows this slot of an automatic pallet changer before and after a data collector has been attached to it.

Plots display the trend of a variable with respect to time. They aid system designers to identify whether steady–state simulation has been achieved and are attached to slots during this stage of the model development process. For example, plots are attached to the STATUS slot of each workstation to display the average utilization of each workstation with respect to time. Plots are also attached to the STATUS slot of the material handling system if either AGVs or rail carts are used

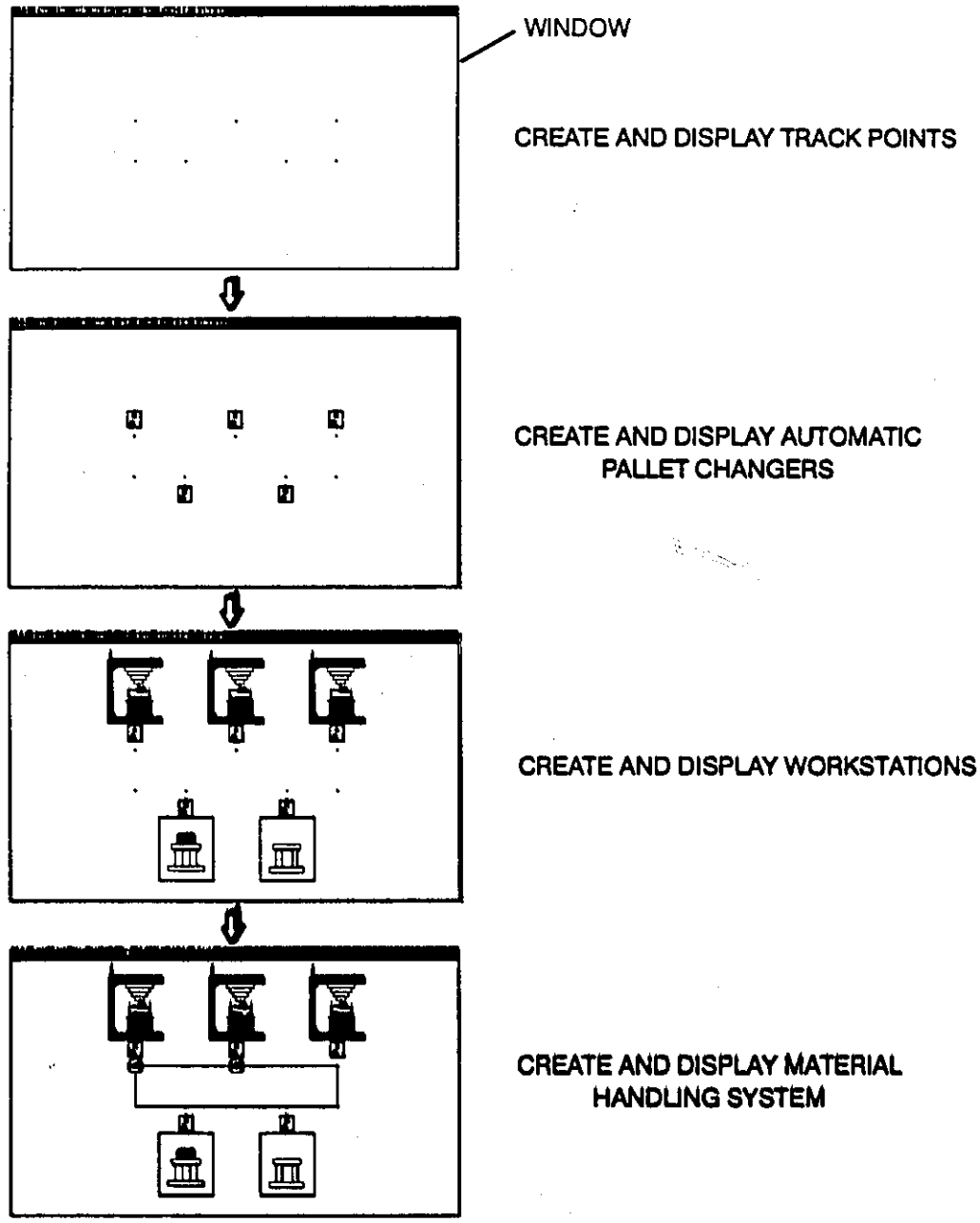


Figure 5.17. Sample displays of the simulation model development

BEFORE ATTACHING DATA COLLECTOR

```
(Output) The NCMM.A.2 Unit in FMS.DESIGN Kn
Member slot: STATUS from QLIB.COMPONENTS
   Inheritance: OVERRIDE.VALUES
   ValueClass:
               (ONE.OF BUSY
                       IDLE
                       FAILED
                       BLOCKED)
   Cardinality.Max: 1
   Cardinality.Min: 1
   Comment: "current status of this component"
   Values: IDLE
```

AFTER ATTACHING DATA COLLECTOR

```
(Output) The NCMM.A.2 Unit in FMSMOD Knowled
Own slot: STATUS from NCMM.A.2
   Inheritance: OVERRIDE.VALUES
   ValueClass:
               (ONE.OF BUSY
                       IDLE
                       FAILED
                       BLOCKED)
   Avunits: DC.FOR.STATUS.OF.NCMM.A.2,
            UTILIZATION.PLOT.AV in FMSLIB
   Cardinality.Max: 1
   Cardinality.Min: 1
   Comment: "current status of this component"
   Data.Collectors: DC.FOR.STATUS.OF.NCMM.A.2
   Values: IDLE
```

Figure 5.18. Attaching a data collector to the STATUS slot

to transport loaded and empty pallets. Plots showing the number of parts of each part type that are finished with respect to time are also attached to slots. A typical display of a plot attached to the STATUS slot of a workstation is shown in Figure 5.20.

5.5 Model execution

Model execution involves running the simulation model of the system on a computer and generating output data as the simulation progresses. This output

BEFORE ATTACHING DATA COLLECTOR

```
(Output) The APC.2.B.1 Unit in FMS.DESIGN Knowledge Base
Member slot: NUMBER.FOR.WKS from AUTOMATIC.PALLET.CHANGERS
   Inheritance: OVERRIDE.VALUES
   ValueClass: INTEGER
   Cardinality.Max: 1
   Cardinality.Min: 1
   Comment: "The number of parts that have left the apc for the workstation
   Values: UNKNOWN
```

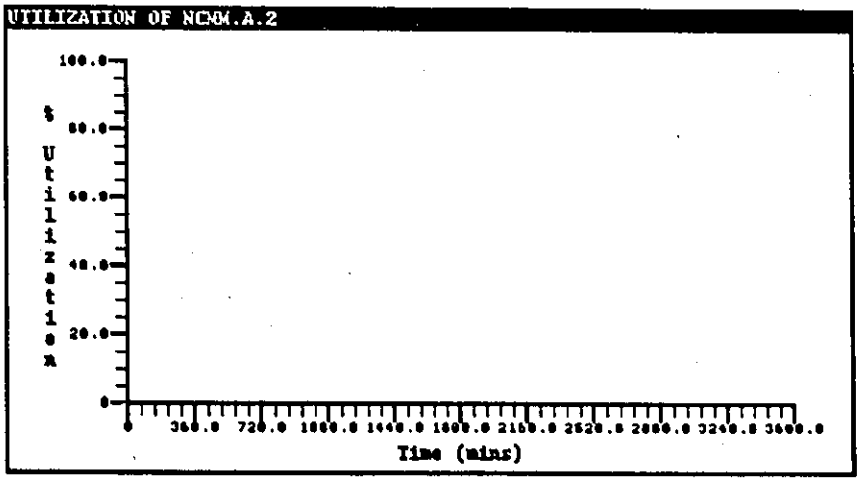
AFTER ATTACHING DATA COLLECTOR

```
(Output) The APC.2.B.1 Unit in FMSMOD Knowledge Base
Own slot: NUMBER.FOR.WKS from APC.2.B.1
   Inheritance: OVERRIDE.VALUES
   ValueClass: INTEGER
   Avunits: DC.FOR.NUMBER.FOR.WKS.OF.APC.2.B.1
   Cardinality.Max: 1
   Cardinality.Min: 1
   Comment: "The number of parts that have left the apc for the workstation"
   Data.Collectors: DC.FOR.NUMBER.FOR.WKS.OF.APC.2.B.1
   Values: UNKNOWN
```

Figure 5.19. Attaching a data collector to the NUMBER.FOR.WKS slot

data is then analyzed to evaluate the performance of the system. A simulation is a dynamic model of a system's behavior as it changes over time. It generally starts from time zero and stops when the simulated time is greater than or equal to a specified time. Simulation is also a collection of events and the two most important mechanisms for scheduling and triggering discrete events are the clock and calendar. An event represents something that takes place and simulation progresses from one event to another. The time at which it happens is scheduled on the calendar which keeps track of all events and the time at which each event should take place. The clock keeps track of the simulated time and when it reaches the time scheduled for an event, the event takes place. Events can schedule other events

BEFORE SIMULATION



AFTER SIMULATION

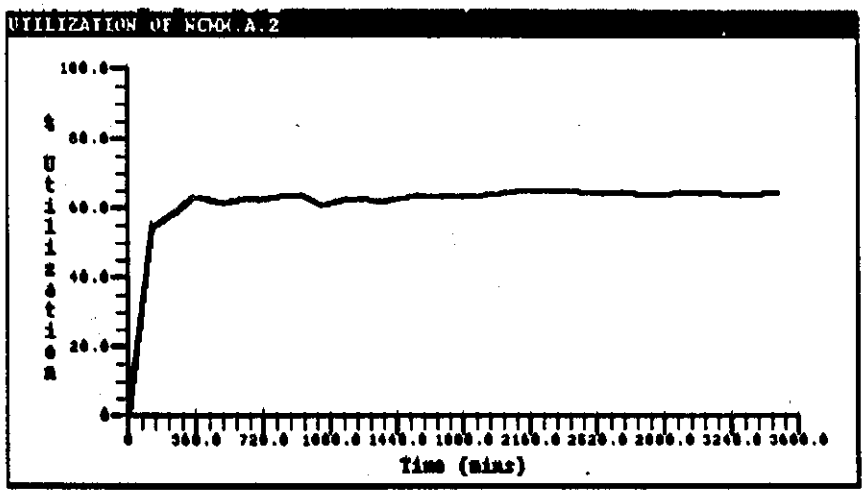


Figure 5.20. Plot for displaying the average utilization of a workstation

to be executed at the same time or at a later time. New events thus generated are added to the calendar. The following issues must be addressed during any simulation study:

(i) ***Initialization:*** Before simulating the model of a flexible manufacturing

system design, several slots of objects in the model have to be initialized. This is to clear all the data that have been collected in the previous run and to reset the values of slots to default values. For example, the slot which contains the average utilization of an AGV is reset to zero and the slot that contains the state of a workstation is reset to "idle".

(ii) ***Animation:*** Computer animation of the model as the simulation progresses is a feature that helps in understanding the actual system and also in verifying the model. A major drawback of animation is the increase in computer time to run the same simulation model.

(iii) ***Post−simulation calculations:*** These are calculations that are done after the simulation is over such as calculating the average utilization of a workstation, the percentage of time a workstation is blocked, the average time parts wait in an automatic pallet changer for a workstation, and the investment cost of the system components.

5.6 Implementation of the simulator

The simulator was developed using the object−oriented programming paradigm of KEE and SimKit. The initialization of a model of a flexible manufacturing system is done by sending a message to the INITIALIZE.MODEL! method slot of the child of SIMULATORS unit in the model knowledge base. This method sends messages to the INITIALIZE! method slot of each object in the model. The value of the INITIALIZE! method slot of each object determines how that particular object will be initialized. For example, the value of the INITIALIZE! method slot for the PALLET.A object in the FMSLIB library is different than that of the AGV.B object (Figure 5.21).

PALLET.A

```
(Output) The PALLET.A Unit in FMSLIB Knowledge Base
Member slot: INITIALIZE! from PALLETS
   Inheritance: METHOD
   ValueClass: METHOD
   Values: INITIALIZE.PALLETS
```

AGV.B

```
(Output) The AGV.B Unit in FMSLIB Knowledge Base
Member slot: INITIALIZE! from TYPE.1.MHS
   Inheritance: METHOD
   ValueClass: METHOD
   Values: INITIALIZE.TYPE.1.MHS
```

Figure 5.21. Value of the INITIALIZE! method slot for different objects

The CALCULATION.RULES rule class has a subclass AVERAGE.VALUE.RULES to calculate average values after simulation and a subclass INVESTMENT.COST.RULES to calculate the investment for each type of component in the system (Figure 5.22). For example the rule unit APC.CALCULATION.RULE calls method slots to calculate the average queue length and the average waiting time for each automatic pallet changer. The TellAndAsk syntax of this rule is shown below:

```
(APC.CALCULATION.RULE
   (IF   (ALL ?APC.TYPE ARE AUTOMATIC.PALLET.CHANGERS)
         (?APC IS IN CLASS ?APC.TYPE)
         THEN
         (LISP (UNITMSG 'EXECUTOR
                        'CALCULATE.AVERAGE.QUEUE.LENGTH! ?APC))
```

(LISP (UNITMSG 'EXECUTOR

'CALCULATE.AVERAGE.WAIT.TIME! ?APC))))

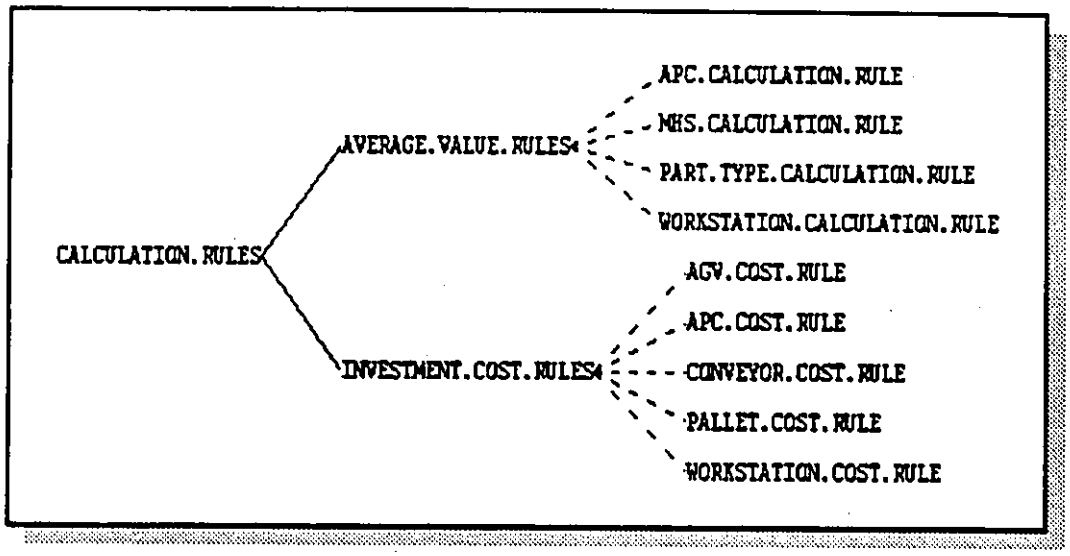


Figure 5.22. CALCULATIONS.RULES rule class

The rule unit APC.COST.RULE, which is a rule unit of INVETSMENT.COST.RULES, calculates the total cost of all automatic pallet changers in the system. The TellAndAsk syntax for this rule is shown below:

(APC.COST.RULE

(IF *(ALL ?APC.TYPE ARE AUTOMATIC.PALLET.CHANGERS)*

(?APC IS IN CLASS ?APC.TYPE)

(THE COST OF ?APC IS ?APC.COST)

THEN

(LISP (PUT.VALUE 'EXECUTOR 'APC.COST

(+ (GET.VALUE 'EXECUTOR 'APC.COST)

?APC.COST)))))

The simulation is animated only if required. The user is provided with the option to turn on or turn off the animation before starting the simulation of the model. Although animation is a good technique to graphically view the dynamics of the system, it takes a significant amount of computer time to run and should be used only when absolutely necessary.

5.7 Discussion

The output from the design synthesizer is a list of selected system components, the relationships between them, and their approximate positions in the layout. The traditional approach to understand the dynamics of this new design is to develop a simulation model of the design using a simulation language. Even with the current generation of user friendly simulation languages provided with powerful graphic capabilities, it takes a considerable amount of expertise and time to develop a simulation model of a complex system such as a flexible manufacturing system.

The simulation model developer module of FMX eliminates this time–consuming manual process by automatically developing graphical simulation models. Thus, it significantly reduces programming time and effort required to develop models. This module establishes the relationships between the system components, develops a graphical display of the model using icons, attaches data collectors to the various components for collecting data, and attaches plots to data collectors for plotting values of variables over time. The object–oriented approach facilitates simulation model development by sending messages to the appropriate method slots. Once the model is developed, the simulator module of FMX executes

the model by initializing it, running the model and animating the display, and calculating average values at the end of the simulation run.

CHAPTER 6

ANALYZER

This chapter begins with a brief description of the simulation output analysis process and proceeds to outline the functions of the expert analyzer developed in this thesis. It then describes the various measures which are used to evaluate the performance of flexible manufacturing systems. It also discusses the strategy for identifying deficiencies in the FMS designs and recommending changes to overcome these deficiencies. The use of conflict resolution strategies when more than one design change is recommended is explained in detail. Finally, this chapter presents the implementation of the analyzer module of the FMX prototype using KEE.

6.1 Simulation output analysis

FMS designers peruse enormous output data, both numeric and graphic, which is generated after every simulation run to evaluate the performance of a design. They use performance measures for this purpose. These performance measures are quantifiable variables which act as guidelines for evaluating FMS designs. Their values are set prior to the design synthesis phase and forms the basis for accepting a given design or recommending changes to improve its performance.

Performance measures that are commonly used for analyzing simulation outputs are:

(i) part production rates,

(ii) average equipment utilizations,

(iii) average equipment blocking, and

(iv) average queue lengths of buffers.

After each simulation, values of these performance measures are compared with the preset values to check the goodness of a design.

A system design which satisfies all specified performance measures is considered to be a good design and is recommended for implementation. However, if the performance measures are not satisfied, system designers use their knowledge and experience to identify deficiencies in that design and recommend changes to overcome them. They then implement these changes by developing a simulation model of the new design and simulating it. Thus the original system design is constantly improved through several iterations. This, however, is a time–consuming process and consequently, increases the FMS design cycle time.

6.2 Functions of the analyzer

One of the objectives of this research was to automate the output analysis process by capturing the knowledge of the system designers in an expert system. The analyzer, developed in this thesis, is a diagnostic expert system that contains the knowledge and experience of system designers to perform the following three major functions as shown in Figure 6.1:

(i) ***Comparison:*** It compares the values of the various performance measures obtained from the simulation model with the preset values to see if they

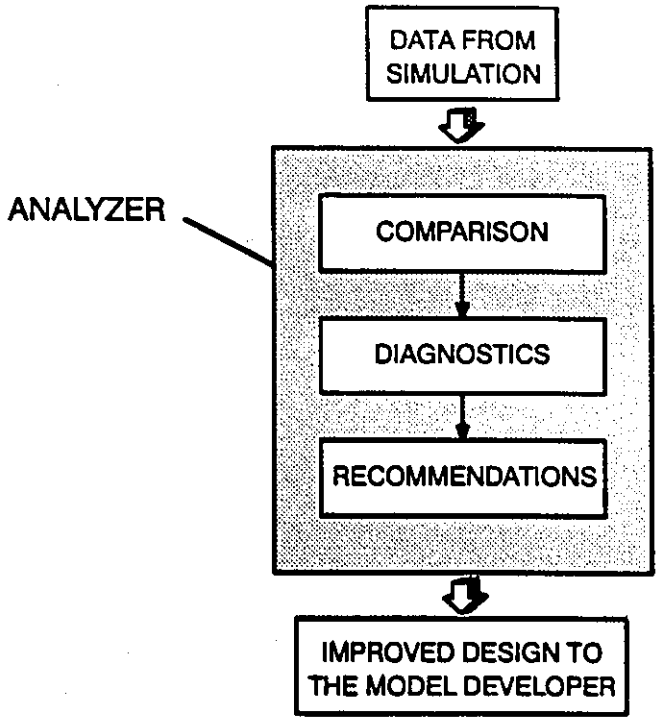


Figure 6.1. Major functions of the expert analyzer

are satisfied.

(ii) ***Diagnostics:*** If the specified performance measures are not satisfied, it diagnoses the problems and identifies design deficiencies in the current system design.

(iii) ***Recommendations:*** It recommends design changes to overcome the identified design deficiencies and implements these changes by passing the improved design data back to the simulation model developer.

Effectively, the analyzer replaces the human being from the design loop, thereby automating the entire FMS design process. Because this design process is iterative, the analyzer performs these functions at the end of each iteration after an FMS

design has been simulated. The iterations stop when a particular design satisfies all its performance measures.

6.3 Performance measures

Performance measures, as mentioned earlier in Section 6.1, are quantifiable variables that are used to evaluate FMS designs. The acceptable values of these performance measures are set prior to design synthesis by the user and the values from the simulation model are compared with the preset values at the end of each simulation run. In this research, the following performance measures were considered for evaluating an FMS design:

(i) production volume of all part types in the family,

(ii) utilizations of workstations and material handling systems such as AGVs or rail carts,

(iii) blocking at workstations, and

(iv) queue lengths of automatic pallet changers.

6.3.1 Production volumes of part types

A flexible manufacturing system should have sufficient capacity to produce a family of part types at the required production rates. For example, let a family consist of three part types, A, B, and C with targeted production volumes of V_a, V_b, and V_c, within a period T. An FMS designed for these three part types should produce the required volumes, V_a, V_b, and V_c, in time T.

It is, however, practically impossible to exactly meet specified targeted production volumes. Therefore, an acceptable production volume, which is a percentage of the targeted production volume, is defined and is set for each part

type in the family. This acceptable production volume for each part type in the family is calculated using equation (6.1).

For each part type,

*Acceptable production volume = % acceptable * targeted production volume* (6.1)

An FMS design satisfies this performance measure if the production volume of each part type from the simulation model is greater than or equal to the targeted production volume as shown in equation (6.2).

$$VS_i \geq VA_i \qquad \text{for all i} \qquad (6.2)$$

where,

VA_i – Acceptable production volume for part type i

VS_i – Production volume of part type i from simulation model

For example, if the targeted production volume for a part type is 200 units and the acceptable percentage is 95%, then the production volume for this part type obtained at the end of a simulation run should be as shown below:

$$VS \geq 190$$

6.3.2 Utilization of workstations and material handling systems

Utilization of a component is defined as the percentage of time the component is busy to the total simulated time. The expected utilizations of the workstations and material handling systems, such as AGVs or rail carts, are always specified as a range with a higher and a lower limit. The average utilization of each

workstation type or material handling system type obtained at the end of a simulation run should lie within this range as shown in equation (6.3).

$$U_{LLi} \leq U_{AVi} \leq U_{ULi} \qquad \text{for all i} \qquad (6.3)$$

where,

U_{ULi} – Upper limit of expected utilization of workstation or material handling system type i

U_{LLi} – Lower limit of expected utilization of workstation or material handling system type i

U_{AVi} – Average utilization of workstation or material handling system type i obtained from the simulation model

For example, if the expected utilization of a workstation type is set between 60% and 70%, then the average utilization of this workstation type obtained at the end of a simulation run should be as shown below:

$$60\% \leq U_{AV} \leq 70\%$$

6.3.3 Blocking of workstations

The phenomenon of blocking was explained earlier in Section 4.1.2. A blocked workstation has a ripple effect on all components feeding parts to it. The upstream components can no longer send parts to the blocked workstations and ultimately get blocked themselves. This consequently leads to a decrease in the throughput of the system.

Workstation blocking is always expressed as a percentage and theoretically

a system should be designed with absolutely no blocking or zero percentage blocking. This, however, is practically not possible due to the complex interactions in an FMS. Therefore, an allowable blocking percentage for each workstation is specified initially. The percentage blocking of a workstation type obtained at the end of a simulation run should be less than this specified percentage as shown in equation (6.4).

For each workstation type,

$$B_{Si} \leq B_{Ai} \qquad \text{for all i} \qquad (6.4)$$

where,

B_{Ai} – Percentage blocking allowed for a workstation type i

B_{Si} – Percentage blocking of workstation type i from simulation model

For example, if the allowable percentage blocking time of a workstation type is 10%, then actual percentage of time the workstation is blocked obtained at the end of a simulation run should be as shown below:

$$B_S \leq 10\%$$

6.3.4 Queue length of automatic pallet changers

Automatic pallet changers in an FMS act as buffers for storing both loaded and empty pallets at workstations. The maximum number of pallets that could be mounted on them represents their capacity and when an automatic pallet changer is filled to capacity, half of the pallets are waiting for the workstation while the other half are waiting for the material handling system. This finite capacity of the automatic pallet changer often constrains the system when it is filled to capacity.

The average queue length of an automatic pallet changer is a good indicator of its capacity and it is divided into the following:

(i) average queue length of pallets waiting for the workstation, and

(ii) average queue length of the pallets waiting for the material handling system.

If the average queue length of pallets waiting for the material handling system is equal to half the capacity of the automatic pallet changers, the respective workstations get blocked and there is a corresponding increase in the blocking percentage. This increases the average number of pallets waiting for a workstation and ultimately impacts upstream workstations that send parts to this automatic pallet changer.

Therefore, the average queue length of pallets waiting for the workstation or the average queue length of parts waiting for the material handling system is a good indicator of its capacity. A percentage of half the capacity of the automatic pallet changer is specified as the average queue length limit that should not be exceeded. If the average queue length of those pallets waiting for the workstation or waiting for the material handling system, obtained at the end of a simulation run, is less than this value as shown in equation (6.5) or equation (6.6), then the automatic pallet changer has sufficient capacity.

For each automatic pallet changer,

$$Q_{WAVi} \leq \frac{A_p * C_i}{2} \qquad \text{for all i} \qquad (6.5)$$

$$Q_{MHAVi} \leq \frac{A_p * C_i}{2} \qquad \text{for all i} \qquad (6.6)$$

where,

Q_{WAVi} – Average queue length of parts waiting for workstation at automatic pallet changer i obtained from the simulation model

Q_{MHAVi} – Average queue length of parts waiting for material handling system at automatic pallet changer i obtained from the simulation model

A_P – Allowable queue length percentage

C_i – Capacity of automatic pallet change i

For example, if the capacity of an automatic pallet changer is 4 and the allowable percentage is 75%, then the average queue length of parts waiting for the workstation or material handling system obtained at the end of a simulation run should be as shown below:

$$Q_{WAVi} \leq \frac{75 * 4}{100 * 2} \qquad \text{for all i}$$

$$Q_{MHAVi} \leq \frac{75 * 4}{100 * 2} \qquad \text{for all i}$$

6.4 Identifying deficiencies and recommending changes

The various performance measures and the acceptable regions for the values of these performance measures obtained from the simulation model are shown in Figure 6.2. For example, the acceptable region for the average utilization of a workstation type is between L_L and U_L as shown by the vertical lines in the figure. Any FMS design whose performance measures fall in the respective acceptable regions is recommended as the final design and the FMS design process is terminated. However, a current design whose performance measure(s) fall in the unacceptable regions has deficiencies that have to be identified first. After

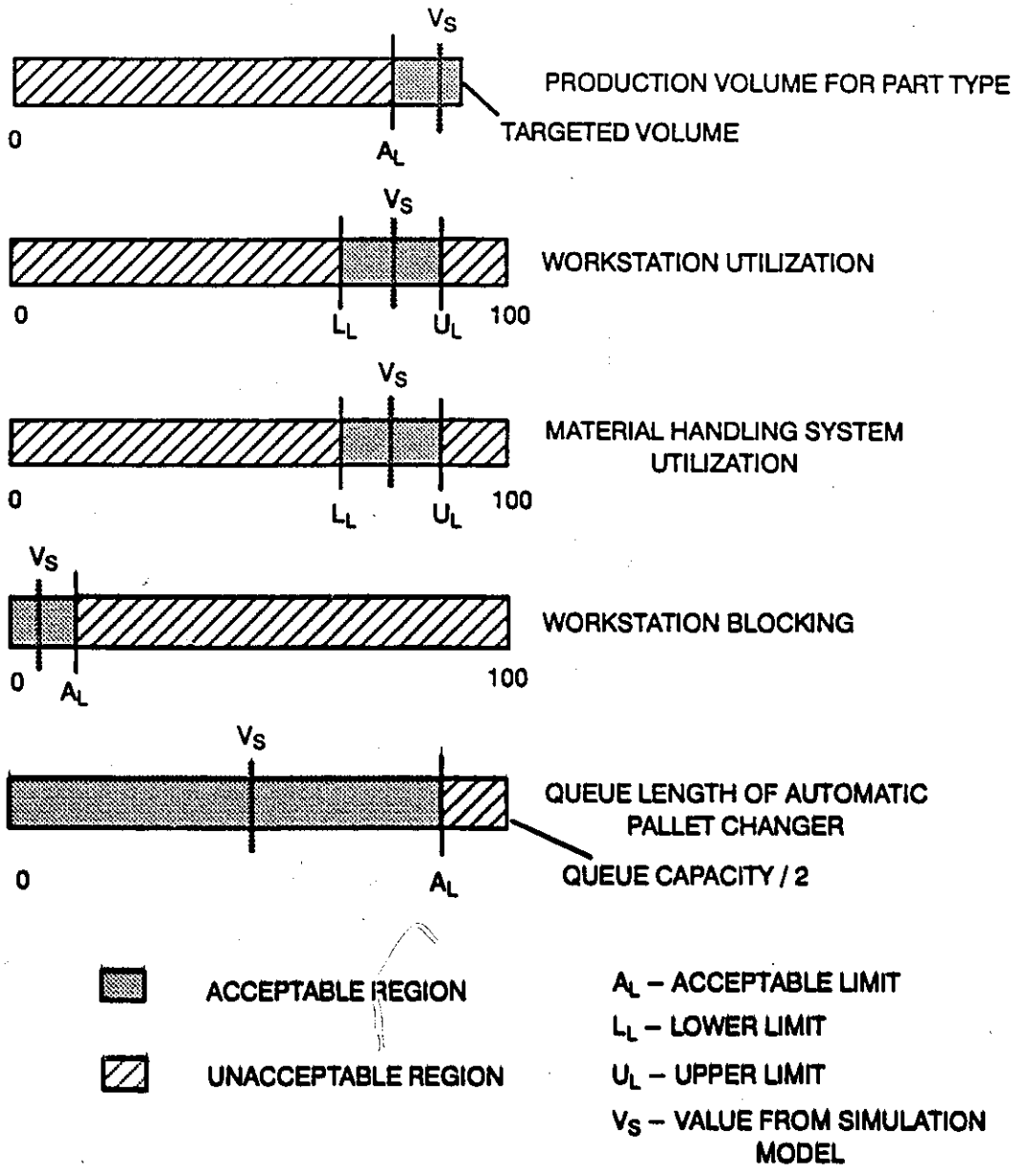


Figure 6.2. Acceptable regions for performance measures

identifying these deficiencies, the design changes that could be made are recommended and the changes implemented.

In systems as complex as the FMS, system designers use heuristics to

compare the values of various performance measures, identify design deficiencies, recommend changes to overcome these deficiencies, and implement these changes. Often more than one course of action might be taken to overcome a deficiency. In such situations, conflict resolution strategies are used to select the appropriate recommendation. The following sections explain in detail the most commonly encountered design problems and the changes that could be made to eliminate these problems.

6.4.1 Under–produced part types

A part type is under–produced if the FMS is unable to produce the acceptable volume of this part type. The production volume of each part type at the end of the simulation is compared with the acceptable production volume to determine the part types that are under–produced. Examples of rules for this purpose are shown below:

IF the targeted production volume of a part type is TP,
and the acceptable percentage is A
THEN the acceptable production volume of this part type is A * TP / 100.

IF the production volume of a part type, obtained at the end of a simulation run, is less than the acceptable production volume
THEN this part type is under–produced.

The number of each pallet type needed to maintain a steady flow of parts in and out of the system was determined during design synthesis. If the number of a particular pallet type is lower than expected it reduces the number of parts that

this pallet type could carry. Hence, there is a reduction in the number of parts of this part type circulating in the system. This lowers the production volume of that part type.

The production rate of a under–produced part type can be increased by increasing the number of parts of this part type circulating in the system. This can be done by increasing the number of pallets of the pallet type that carries this under–produced part type.

The amount of time a part spends in the system includes travel time and waiting time. When the travel times and waiting times are long, the production rates of all part types decrease which lowers their production volumes. By decreasing the travel time or waiting time, the production rates can be increased. This is done by either increasing the speed or the number of the material handling system(s) which transport parts.

Example of rules to recommend the changes when part types are under–produced is shown below:

IF a part type is under–produced
THEN increase the number of pallets of the pallet type that carries this part type,
or increase the speed of the material handling system.

IF a part type is under–produced,
and the material handling system is either AGVs or rail carts
THEN increase the number of either AGVs or rail carts.

If the number of pallets of a pallet type is to be increased, the new number

of pallets is calculated based on the current number, current production volume, and the expected production volume as shown in equation (6.7).

$$New\ pallet\ number\ =\ \left\lceil \frac{Current\ number\ *\ Expected\ production\ volume}{Current\ production\ volume} \right\rceil \quad (6.7)$$

If the speed of the material handling system is to be increased, the new travel speed is calculated based on the current travel speed, current production volume, and expected production volume as shown in equation (6.8).

$$New\ travel\ speed\ =\ \frac{Current\ travel\ speed\ *\ Expected\ production\ volume}{Current\ production\ volume} \quad (6.8)$$

If the number of AGVs or rail carts have to be increased, the new number is calculated based on the current number, current production volume, and total production volume as shown in equation (6.9).

$$New\ AGV\ number\ =\ \left\lceil \frac{Current\ number\ *\ Expected\ production\ volume}{Current\ production\ volume} \right\rceil \quad (6.9)$$

The above equations [(6.7), (6.8), and (6.9)] assume linear relationships. Although some non-linearity may exist, they still provide good estimates.

6.4.2 Over-utilized or under-utilized workstations

A flexible manufacturing system consists of several workstation types with a number of one or more for each workstation type. At the end a simulation run, the average utilization of each workstation type is obtained and compared with the established lower and upper limits to see if this workstation type is over-utilized

or under–utilized. Average utilization for workstation types with a number of more than one is calculated using equation (6.10).

For each workstation type i,

$$\textit{Average utilization} = \frac{\sum_{j=1}^{j=W_i} U_{ij}}{W_i} \qquad \text{for all i} \qquad (6.10)$$

where,

U_{ij} – Average utilization of workstation j of type i

W_i – Total number of workstations of type i

Examples of rules for comparing the average utilization of each workstation type with the established limits are shown below:

IF the average utilization of a workstation type is less than the lower limit of the expected utilization range

THEN this workstation type is under–utilized.

IF the average utilization of a workstation type is greater than the lower limit of the expected utilization range and is less than the upper limit of the expected utilization range

THEN this workstation type is correctly utilized.

IF the average utilization of a workstation type is greater than the upper limit of the expected utilization range

THEN this workstation type is over–utilized.

Workstations are over–utilized because there is always a part waiting to be processed or parts that are being processed spend long time on these workstations due to long processing times. These workstations are always busy and reduce the number of parts coming out of the system. They are the constraints or bottlenecks of the system. On the other hand, when there is an excess capacity some workstation types are idle most of the time and are under–utilized. Both these situations are not desirable and have to be eliminated to make the system more efficient.

When a certain workstation type is over–utilized, the course of action is to increase its number. On the other hand, when a certain workstation type is under–utilized, the course of action is to decrease its number. It is, however, not feasible to decrease the number of an under–utilized workstation type by a number of 1 if there is only a single workstation of this type. Example of rules to increase or decrease workstation numbers are shown below:

IF a workstation type is over–utilized
THEN increase the number of this workstation type.

IF a workstation type is under–utilized,
and the number of this workstation type is greater than 1
THEN decrease the number of this workstation type.

The new number for a workstation type is calculated based on the current utilization, expected utilization, and the current number of workstations as shown in equation (6.11).

$$New\ workstation\ number = \left\lceil \frac{Current\ number * Current\ utilization}{Expected\ utilization} \right\rceil \quad (6.11)$$

6.4.3 Over–utilized or under–utilized AGVs and rail carts

When AGVs or rail carts which are used as material handling system, at the end of each simulation, the average utilization of each AGV or rail cart type is obtained and compared with the established lower and upper limits to see if they are over–utilized or under–utilized. Average utilization for AGV or rail cart types with a number of more than one is calculated using equation (6.12).

For each AGV or rail cart type i,

$$Average\ utilization = \frac{\sum_{j=1}^{j=M_i} U_{ij}}{M_i} \qquad \text{for all i} \quad (6.12)$$

where,

U_{ij} – Average utilization of AGV or rail cart j of type i

M_i – Total number of AGVs or rail carts of type i

Examples of rules for comparing the average utilization of each workstation type with the established limits are shown below:

IF the average utilization of a AGV or rail cart type is less than the lower limit of the expected utilization range

THEN this AGV or rail cart type is under–utilized.

IF the average utilization of a AGV or rail cart type is greater than the lower limit of the expected utilization range and is less than the upper limit of the expected utilization range

THEN this AGV or rail cart type is correctly utilized.

IF the average utilization of a AGV or rail cart type is greater than the upper limit of the expected utilization range

THEN this AGV or rail cart type is over–utilized.

AGVs or rail carts are over–utilized because there is always a part waiting to transported. These AGVs or rail carts are always busy and they reduce the number of parts coming out of the system. They are the constraints or the bottlenecks in the system. On the other hand, when there is an excess capacity some AGV or rail cart types are idle most of the time and are under–utilized. Both these situations are not desirable and have to be eliminated to make the system more efficient.

When a certain AGV or rail cart type is over–utilized, the immediate action is to increase its number. On the other hand, when a certain AGV or rail cart type is under–utilized, the course of action is to decrease its number. Increasing the number of AGV or rail cart type is an expensive proposition because of the cost of AGV or rail cart type. Therefore, it is always more cost effective to increase the speed of all AGVs or rail cart types in the system to their maximum allowable speed before considering an increase in the number. Example of rules to increase or decrease AGV or rail cart numbers are shown below:

IF an AGV or rail cart type is over–utilized

THEN increase the number of this AGV or rail cart type.

IF an AGV or rail cart type is under−utilized,
and the number of this AGV or rail cart type is greater than 1

THEN decrease the number of this AGV or rail cart type.

The new number for an AGV or a rail cart type, whether it is an increase or decrease, is calculated based on the current utilization, expected utilization, and the current number of AGVs or rail carts as shown in equation (6.13).

$$New\ AGV\ number = \left\lceil \frac{Current\ number * Current\ utilization}{Expected\ utilization} \right\rceil \quad (6.13)$$

These situations, however, do not arise when conveyors are used as the material handling system. This is because as soon as a space is available on the conveyor a loaded or empty pallet gets on it and starts travelling to the next workstation. In addition to the local storage at each workstation, these conveyors act as temporary storage.

6.4.4 Blocking of workstations

Automatic pallet changers in an FMS hold loaded or empty pallets. Loaded pallets contain parts and they wait in an automatic pallet changer to get processed by the workstation or to travel to the next workstation along the part route. Empty pallets, on the other hand, wait in an automatic pallet changer for a material handling system, usually a rail cart or an AGV.

If the rail carts or AGVs are utilized extensively because they are less in number than required, then loaded or empty pallets in an automatic pallet changer which are waiting for a rail cart or AGV prevent a processed part from being

transferred to the automatic pallet changer if it is loaded to capacity. This automatically forces the processed part to stay in the workstation, thereby blocking other parts that are waiting for the workstation. The effect of blocking can be seen in either an increase in the average number of pallets waiting for a workstation, or the average number of parts waiting for the material handling system. These increases are also reflected in the increase in the percentage of time a workstation is blocked. Example of rules that compare the average queue lengths for each automatic pallet changer and the blocking percentage for each workstation in the system are shown below:

IF the percentage of time a workstation is blocked based on the simulation model is greater than the established blocking percentage
THEN the workstation is over−blocked.

IF the allowable percent is AP,
and if the capacity of the automatic pallet changer is C,
and average queue length of pallets waiting for the material handling system is greater than AP * C / 2
THEN the automatic pallet changer has insufficient capacity

IF the allowable percent is AP,
and if the capacity of the automatic pallet changer is C,
and average queue length of pallets waiting for the workstation is greater than AP * C / 2
THEN the automatic pallet changer has insufficient capacity.

At the end of a simulation, if it is found that the automatic pallet changer has

insufficient capacity, then its capacity should be increased to the next level. Example of the rule to do this is shown below:

IF the automatic pallet changer has insufficient capacity

THEN then increase the capacity of this automatic pallet changer to the next level.

6.4.5 Conflict resolution strategies

Conflict resolution strategies are often used when more than one recommendation is made to overcome the design deficiencies of the present system design. Instead of making all the changes recommended immediately, the one that will cost the least as well as have a significant impact on system performance is chosen first. These are essentially heuristic rules which select the appropriate changes to be made to the design when more than one change is recommended.

Consider the situation when the part's production is not met and the workstations are under–utilized. Based on these two independent symptoms, the recommendations would be to increase the number of pallets circulating in the system or increase the speed of the material handling system or increase the number of AGVs or rail carts or decrease the number of workstations in the system. While by decreasing the number of workstations in the system we can increase the workstation utilization, there is still the same number of pallets circulating in the system as before and the part production rates will still not be satisfied. However, by increasing the number of pallets circulating in the system we not only increase the throughput of the system, but also the utilization of the existing number of workstations because more parts are visiting the same number of workstations. Increasing the number of pallets increases the investment cost. On the other hand,

increasing the speed of the material handling system would reduce the travel and waiting time for parts. Thus, more parts can be produced by the system. Moreover, changing the speed of the material handling system does not incur any additional capital investment. Therefore, given these alternatives, the conflict resolution rule will recommend an increase in the speed of the material handling system.

Consider the situation where the AGVs or carts that transport pallets in the FMS are a bottleneck. In addition to the AGVs being over–utilized, some of the automatic pallet changers in the system are filled to capacity most of the time as indicated by the average number of pallets waiting in them for the material handling system or workstation that is blocked. The possible recommendations to make in such a situation are to increase the speed of AGVs, increase the number of AGVs, or increase the capacity of the automatic pallet changer. While the increase in the capacity of the automatic pallet changer will decrease the average queue lengths of waiting pallets, it would not increase the throughput of the system because the AGVs are still a bottleneck. Of course, blocking of the workstation decreases considerably. An increase in the speed of AGVs or an increase in the number of AGVs will increase the number of pallets circulating in the system and will also move pallets quickly from the automatic pallet changers thus reducing the average queue length. However, increasing the speed of the cart would be recommended because it does not involve any additional capital investment. Therefore, a conflict resolution rule in this situation will increase the speed of the AGVs. However, if the AGVs are already traveling at maximum speed, an increase in the number of AGVs will be recommended.

When workstations are blocked, they impact the average number of pallets waiting at them and also decrease the throughput of the system. Therefore, the two recommendations are to either increase the number of pallets or to increase the

capacity of the automatic pallet changer. Increasing the number of pallets in such a situation will not relieve the blocked workstation. Only an increase in the capacity of the automatic pallet changer will relieve the blocked workstation. Therefore, a conflict resolution rule will recommend an increase in the capacity of the automatic pallet changer. Examples of some of the conflict resolution rules are shown below:

IF the number of pallets of a particular pallet type is to be increased,
and the speed of AGVs is to be increased,
and the number of AGVs is to be increased
THEN increase the speed of AGVs in the new design.

IF the capacity of an automatic pallet changer is to be increased,
and the speed of AGVs is to be increased,
and the number of AGVs is to be increased
THEN increase the speed of AGVs in the new design.

IF the number of pallets of a pallet type is to be increased,
and the capacity of an automatic pallet changer is to be increased
THEN increase the capacity of the automatic pallet changer to the next level in the new design.

6.5 Implementation of the analyzer

The expert analyzer contains both heuristic and analytical knowledge that is necessary for analyzing simulation output, identifying deficiencies in the current design, recommending changes to overcome these deficiencies, and implementing these changes for further simulation. In knowledge-based systems heuristic

knowledge is represented as rules while analytical knowledge is represented as methods. In KEE, these rules are organized into rule classes and are written using the TellAndAsk syntax while the methods are written in LISP.

6.5.1 Analyzer module of FMX

The analyzer module of the FMX prototype consists of rules and rule classes similar to the design synthesizer module. The main rule class of this module is FMS.ANALYZER.RULES which in turn consists of four rule classes:

(i) DESIGN.CHANGE.RULES.

(ii) DIAGNOSTIC.RULES,

(iii) GET.INFORMATION.RULES, and

(iv) RECOMMENDATION.RULES.

These rule classes and their hierarchy are shown in Figure 6.3. Rules in the rule

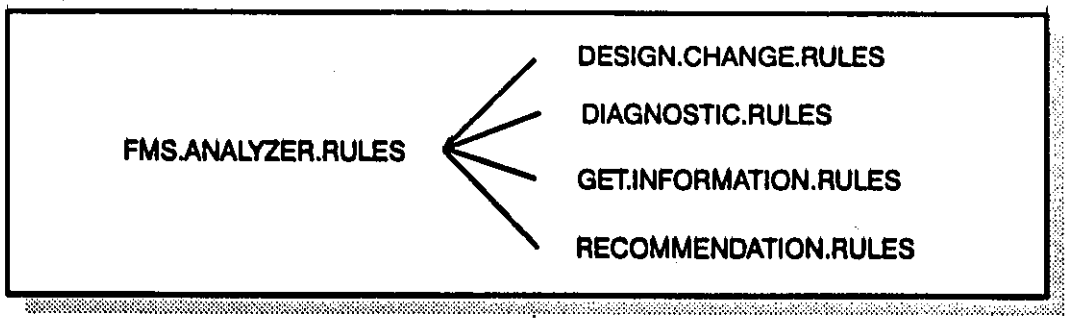


Figure 6.3. Main rule classes in the analyzer

class DIAGNOSTIC.RULES are fired first, followed by rules in the rule class RECOMMENDATION.RULES, and finally rules in the rule class DESIGN.CHANGE.RULES. Rules in the rule class

GET.INFORMATION.RULES are fired indirectly by backward chaining by the rules in the rule class DIAGNOSTIC.RULES.

6.5.1.1 Rules for diagnosis

The rule classes and rules of the rule class DIAGNOSTIC.RULES are shown in Figure 6.4. All the rules belonging to this rule class analyze the output from the simulation models after each run and identify design deficiencies, if present. This rule class is divided into a number of subclasses where each subclass contains rules for diagnosing the problem of each component such as automatic pallet changers, workstations, etc. These rules use a backward chaining approach to get the appropriate information from the rules in the GET.INFORMATION.RULES rule class shown in Figure 6.5. For example, the rules in the rule class BUFFER.CAPACITY.QUANTITY.RULES backward chain on the rules class BUFFER.CAPACITY.CHANGE.RULES which in turn backward chain on the rules class APC.INFORMATION.RULES. The TellAndAsk syntax of the EXCESS.QUEUE.LENGTH.RULE.1, which is a unit of the rule class APC.INFORMATION.RULES, is shown below:

```
(EXCESS.QUEUE.LENGTH.RULE.1
   (IF   (THE ACCEPTABLE.QUEUE.LENGTH OF SYSTEM.DETAILS IS
            ?QUEUE.PERCENT)
         (THE AVERAGE.QUEUE.LENGTH.FOR.MHS OF ?APC IS
            ?AVERAGE.LENGTH)
         (THE CAPACITY OF ?APC IS ?CAPACITY)
         (LISP (>= (/ ?AVERAGE.LENGTH (/ ?CAPACITY 2.0))
                   (/ ?QUEUE.PERCENT 100.0)))
```

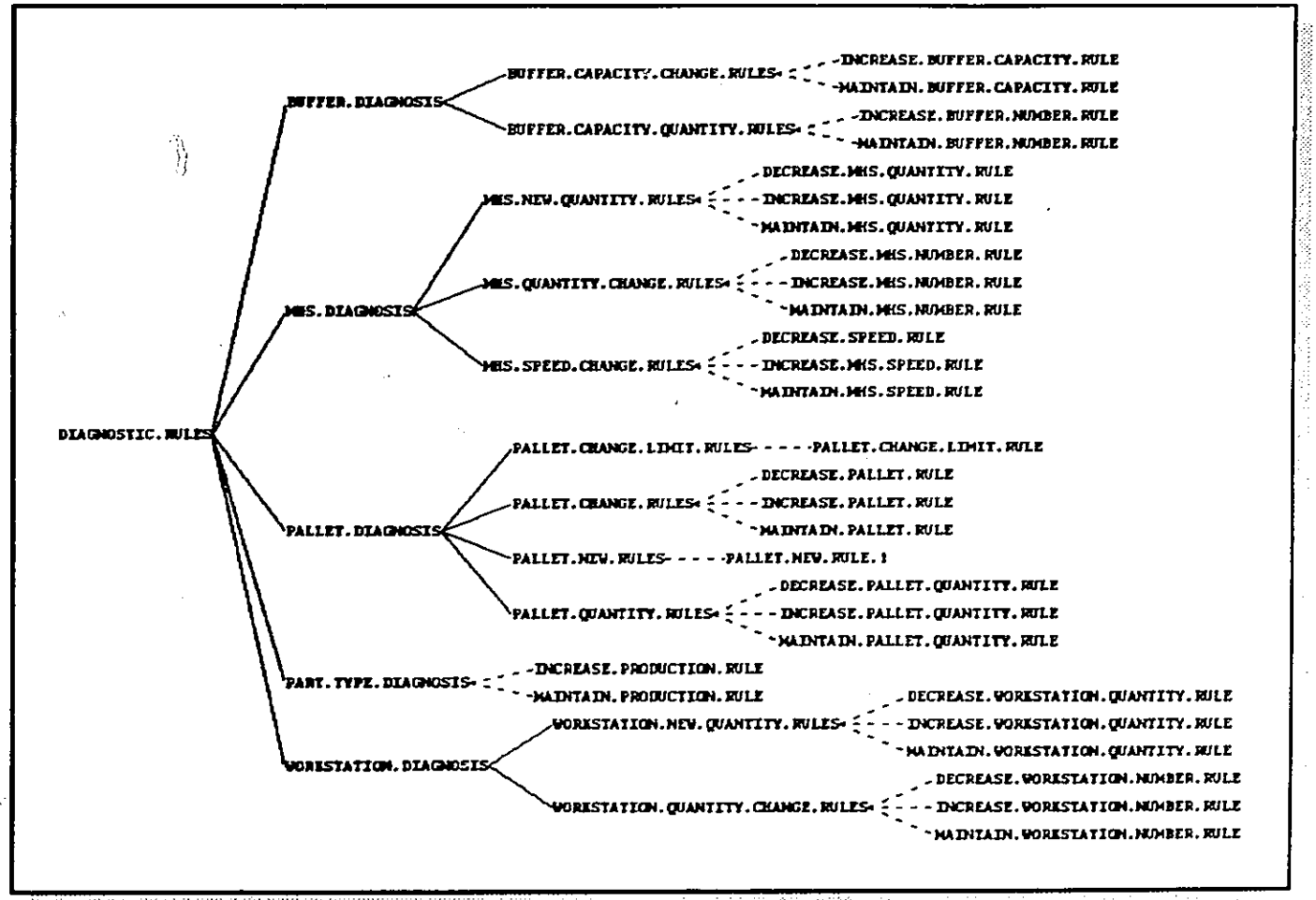


Figure 6.4. DIAGNOSTIC.RULES rule class

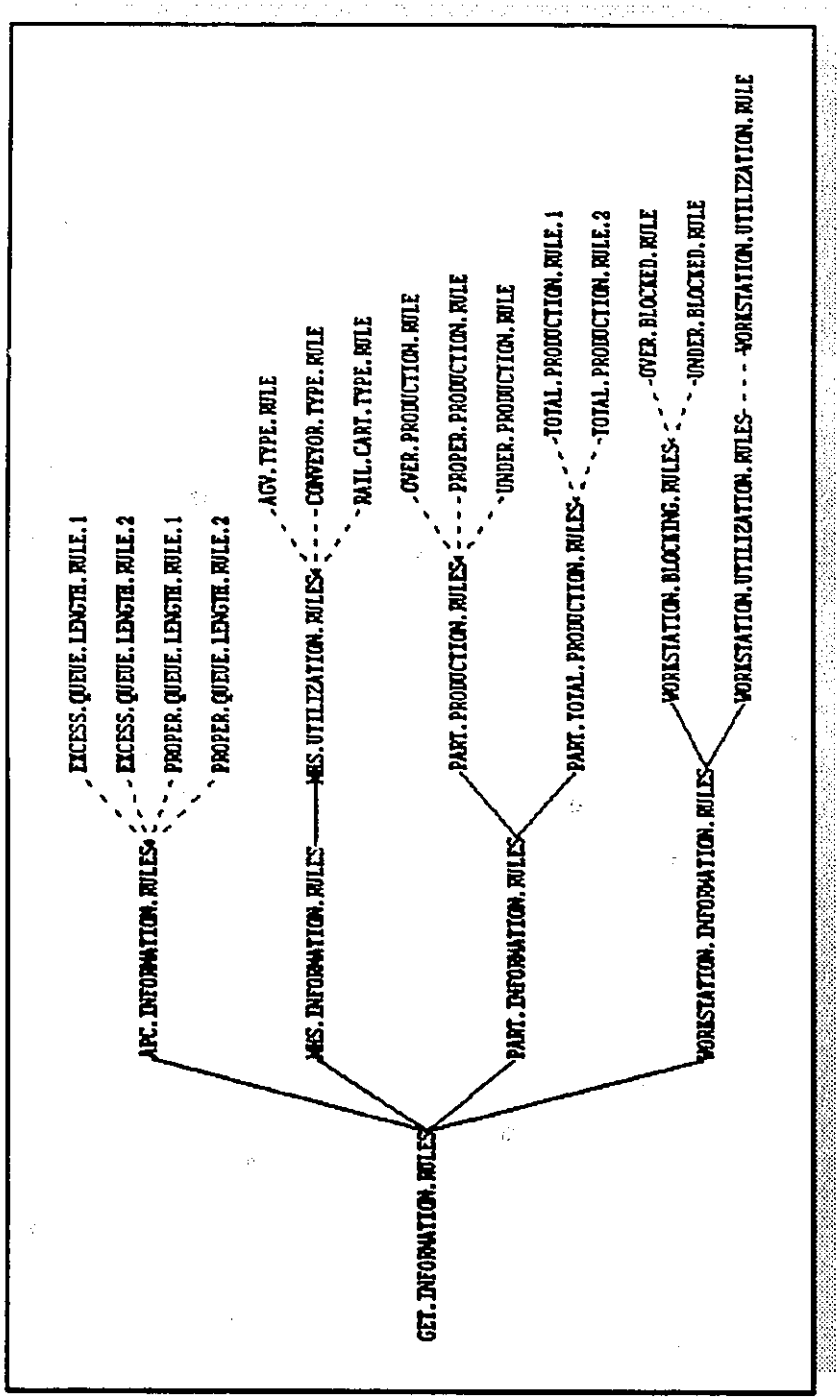


Figure 6.5. GET.INFORMATION.RULES rule class

THEN

(THE AVERAGE.QUEUE.LENGTH.EXCEEDED.FOR.MHS OF ?APC IS YES)))

Firing the rules in this main rule class identifies the deficiencies, if there are any, in the current design. The next set of rules recommend the changes to overcome these deficiencies.

6.5.1.2 Rules for recommendation

The rules belonging to the rule class RECOMMENDATION.RULES are shown in Figure 6.6 are classified into the following two rule classes:

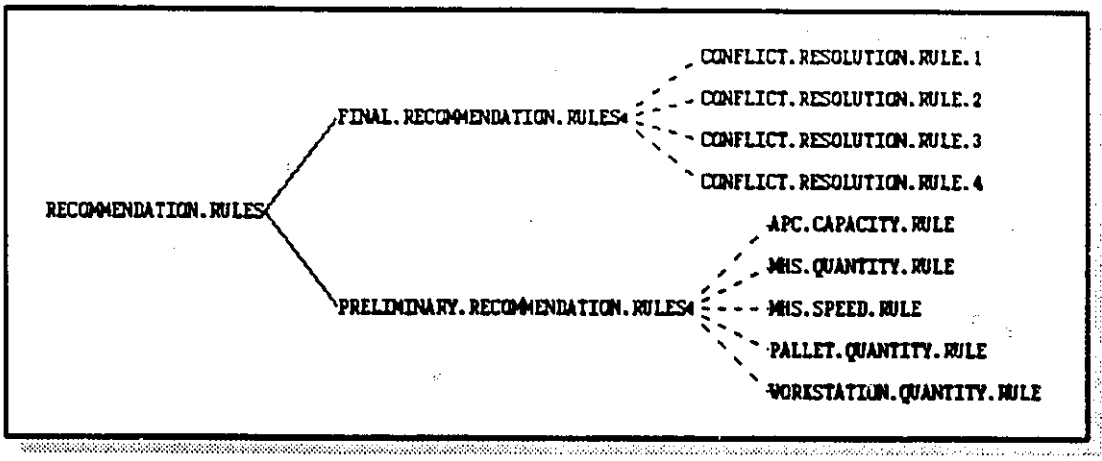


Figure 6.6. RECOMMENDATION.RULES rule class

PRELIMINARY.RECOMMENDATION.RULES which contains rules to recommend changes to the current design for overcoming the design deficiencies, and FINAL.RECOMMENDATION.RULES which contains the conflict resolution rules for selecting a recommendation if more than one recommendation is provided at the preliminary stage. All the rules in the rule class,

PRELIMINARY.RECOMMENDATION.RULES, are fired first followed by the rules in the rule class, FINAL.RECOMMENDATION.RULES, using a forward chaining approach. For example, the rule CONFLICT.RESOLUTION.RULE.3 decides whether the capacity of an automatic pallet changer should be increased or the number of an AGV or a rail cart type is to be increased in the new design when the AGVs or rail carts are traveling at their maximum speed. The TellAndAsk syntax for this rule is shown below:

```
(CONFLICT.RESOULTION.RULE.3
   (IF   (THE APC.CAPACITY.CHANGE? OF DIAGNOSTIC.UNIT IS
           ?APC.CAPACITY)
         (THE MHS.QUANTITY.CHANGE OF DIAGNOSTIC.UNIT IS
           ?MHS.QUANTITY)
         (LISP (EQUAL ?APC.CAPACITY 'YES)
         (LISP (EQUAL ?MHS.QUANTITY 'YES)
         THEN
         (CHANGE.TO (THE MHS.QUANTITY.CHANGE? OF
           DIAGNOSTIC.UNIT IS NO))
         (CHANGE.TO (THE MHS.SPEED.CHANGE? OF
           DIAGNOSTIC.UNIT IS NO))
         (CHANGE.TO (THE WORKSTATION.QUANTITY.CHANGE? OF
           DIAGNOSTIC.UNIT IS NO))
         (CHANGE.TO (THE PALLET.QUANTITY.CHANGE? OF
           DIAGNOSTIC.UNIT IS NO))
```

The rules belonging to this main rule class recommend the change that is to be

made to the current design for the next iteration. The next set of rules uses this recommendation and changes the design accordingly. This change is reflected in the model as well as the graphical display of the model.

6.5.1.3 Rules for design changes

The rules belonging to the rule class DESIGN.CHANGE.RULES are shown in Figure 6.7. Based on the change to be made an appropriate rule is fired.

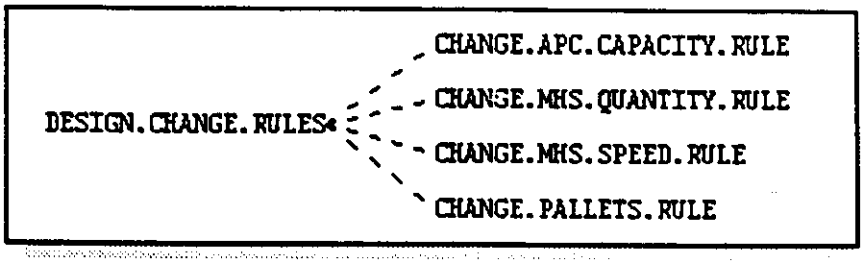


Figure 6.7. DESIGN.CHANGE.RULES rule class

This rule sends a message to a method which in turn messages other methods in the model developer module to make the appropriate changes to the current design. The new design has these changes included in the model as well as the graphical display of the model and is ready for simulation. For example, the rule CHANGE.MHS.QUANTITY.RULE when fired messages the method CHANGE.MHS.QUANTITY! which in turn messages the appropriate methods in the model developer module of FMX for developing a model of the new design. The TellAndAsk syntax of this rule is shown below:

(CHANGE.MHS.QUANTITY.RULE

```
(IF    (THE MHS.QUANTITY.CHANGE? OF DIAGNOSTIC.UNIT IS 'YES)
       THEN
       (LISP (UNITMSG 'ANALYZER.CONTROLLER
                      'CHANGE.MHS.QUANTITY!))))
```

6.6 Discussion

The input to the analyzer is the data from the simulation model after every run. The analyzer is an expert system that analyzes this data to evaluate the performance of the current FMS design. It uses specified performance measures such as part production rates, average equipment utilizations, average equipment blocking and average queue lengths of automatic pallet changers to evaluate designs. The major functions of the analyzer include comparing data from the simulation model with preset values, diagnosing problems and identifying design deficiencies, and recommending design changes. In situations when more than one recommendation is made, the analyzer uses conflict resolution strategies to choose one recommendation that is both cost effective and has a significant impact on the performance of the system.

The analyzer module of FMX contains the knowledge required for analyzing simulation output in the form of rules or methods. Like the design synthesizer, these rules are generic and are organized into various rule classes and rule units. Currently, there are approximately 60 generic rules in the analyzer which use a backward chaining or a goal–oriented strategy to analyze simulation output and recommend design changes. Although these rules are not exhaustive, they consider many important heuristics that are used in practice to change critical design parameters that would improve the performance of flexible manufacturing

systems. The output from the analyzer is a recommended design change which is input to the simulation model developer to generate a model of the new design for the next iteration.

As the initial design is refined, an increase in production is expected after every iteration. However, this increase in production might be due to the addition of a piece of equipment which increases the capital investment cost. This additional investment is justified if there is a significant increase in system throughput. An investment cost index which is the ratio of the total investment to the number of parts produced within a certain payback period was used as a measure to compare the designs in each iteration. This index is calculated automatically at the end of each iteration and a decrease in this index every iteration would suggest that the design changes are increasing the performance of the system.

CHAPTER 7

CASE STUDIES AND DISCUSSION

In this chapter, the use of FMX is illustrated with the help of two case studies. The input information required for each case study is provided. The initial FMS design synthesized by the design synthesizer module of FMX is presented. The responses of the design synthesizer to variation in input information for each case study are presented and discussed. Changes made to the original design by other modules of FMX are discussed in detail for each iteration. Finally, the design which satisfies the specified performance measures for each case study is shown.

7.1 Design synthesis case studies

The potential of the design synthesizer module of FMX is demonstrated with the help of two case studies in this section. The first case study deals with an hypothetical system while the second one deals with an industrial flexible manufacturing system used by a major automotive manufacturer in Ontario. The original data and the results for each case study followed by the response of the design synthesizer to variations in input for these case studies are presented.

7.1.1 Case study 1

Let us consider two part types, BLOCK 1 and BLOCK 2, for which a new

flexible manufacturing system is to be designed. The drawing layout of BLOCK 1 is shown in Figure 7.1 and its process sequence is shown in Figure 7.2. The drawing layout of BLOCK 2 is shown in Figure 7.3 and its process sequence is shown in Figure 7.4. The system requirements are listed in Table 7.1.

The components of the initial system design developed by the design synthesizer are shown in Table 7.2. Based on the dimensions of the part types, PALLET.B was chosen for carrying BLOCK 1 and PALLET.C was chosen for carrying BLOCK 2. Three types of workstations were selected to process both part types—LOAD.A, NCMM.A, and UNLOAD.A. Based on the production volume of both part types and the expected utilization of the workstations, the quantity of each workstation type was determined. One LOAD.A workstation, LOAD.A.1, one UNLOAD.A workstation, UNLOAD.A.1, and three NCMM.A workstations, NCMM.A.1, NCMM.A.2, and NCMM.A.3, were required. The routing for both part types generated by the design synthesizer are shown in Table 7.3. The second and third process in the process sequence for both blocks, given their similarity, were combined by the design synthesizer into a single process which could be performed by either NCMM.A.1, NCMM.A.2, or NCMM.A.3. Thus, alternate routing exists for both blocks and the workstation to which a block is sent during the simulation depends on its availability.

Three numbers of PALLET.B, PALLET.B.1, PALLET.B.2, and PALLET.B.3, and one of PALLET.C, PALLET.C.1, were required. Based on the number of pallets circulating in the system, each workstation is provided with an automatic pallet changer of capacity two. This is the minimum capacity required and five automatic pallet changers each with a capacity of two were selected—APC.2.B.1, APC.2.B.2, APC.2.B.3, APC.2.B.4, and APC.2.B.5.

Both part types visit the same workstation types because they have the same

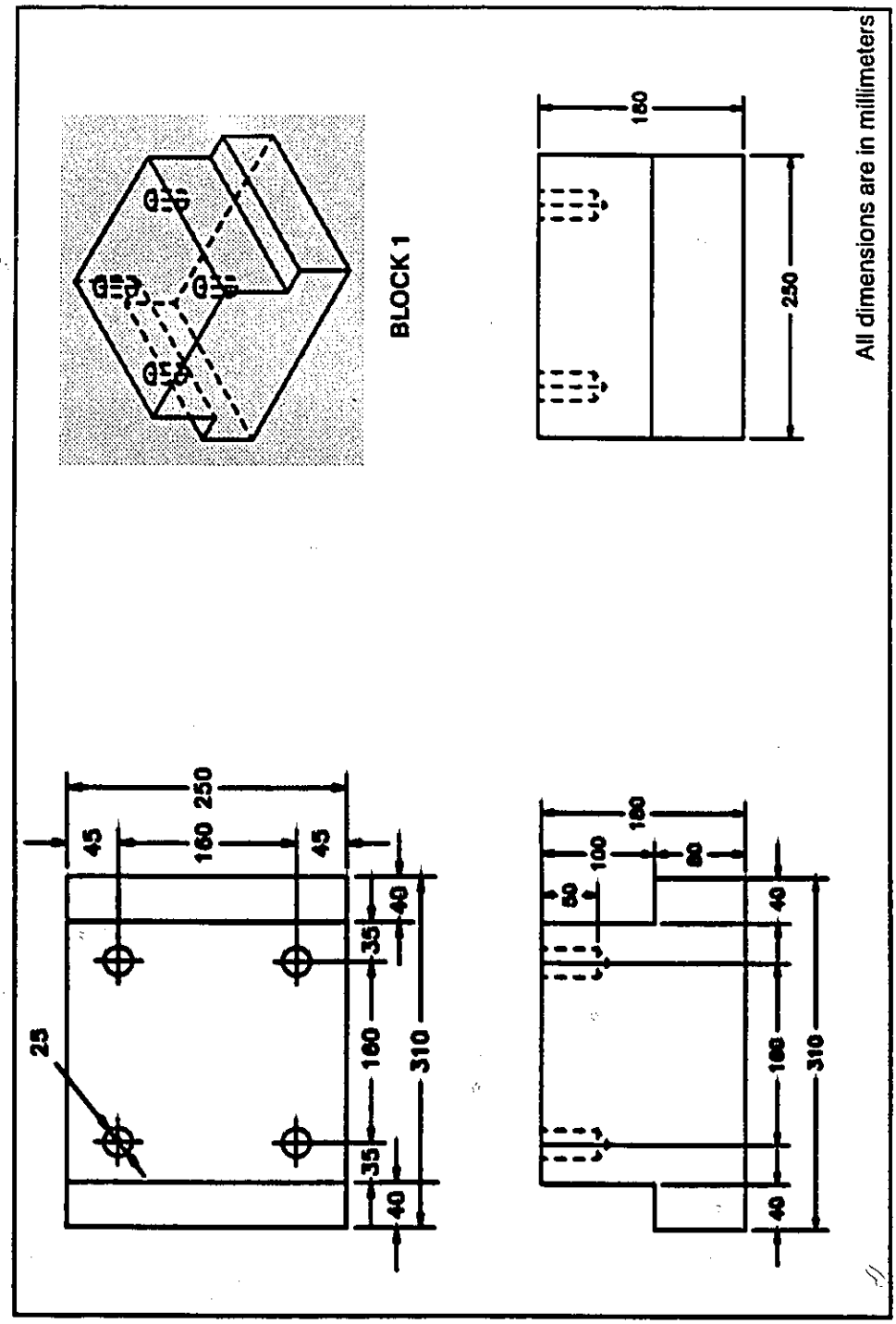
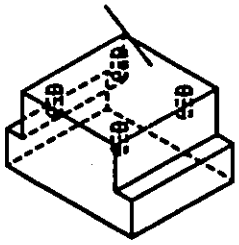


Figure 7.1. Drawing layout of BLOCK 1



BLOCK 1

PROCESS SEQUENCE	PROCESS CODE	SPECIAL TOOLS
Load block on pallet	LOAD–1	NO
Mill surface A	MILL–1	NO
Drill four holes (Φ 25mm) on surface A	DRILL–1	NO
Unload block from pallet	UNLOAD–1	NO

Production volume – 340

Accuracy required – Low

Weight – 140 kgms

Figure 7.2. Process sequence of BLOCK 1

routing. Moreover, they do not require the loading station, LOAD.A.1, for either refixturing or reorienting since this is a serial flow situation. Based on the type of system flexibilities (Table 7.1) and the type of work flow, automated guided vehicles (AGVs) were selected as the material handling system. Based on the maximum dimensions of the pallet type and combined weight of both part type and pallet type, AGV.B was selected and two vehicles were required—AGV.B.1 and AGV.B.2.

Based on the type of material handling system (AGVs), the type of work flow, and the available floor space, a loop type layout was selected. The workstations were laid out on both sides of the loop to minimize the travel distance. The graphical simulation model of this initial flexible manufacturing system design generated by the simulation model developer is shown in Figure 7.5.

To demonstrate the potential of the design synthesizer, the input data of this case study was changed and the corresponding changes in the solutions were observed. Three types of independent changes were made to this case study. These

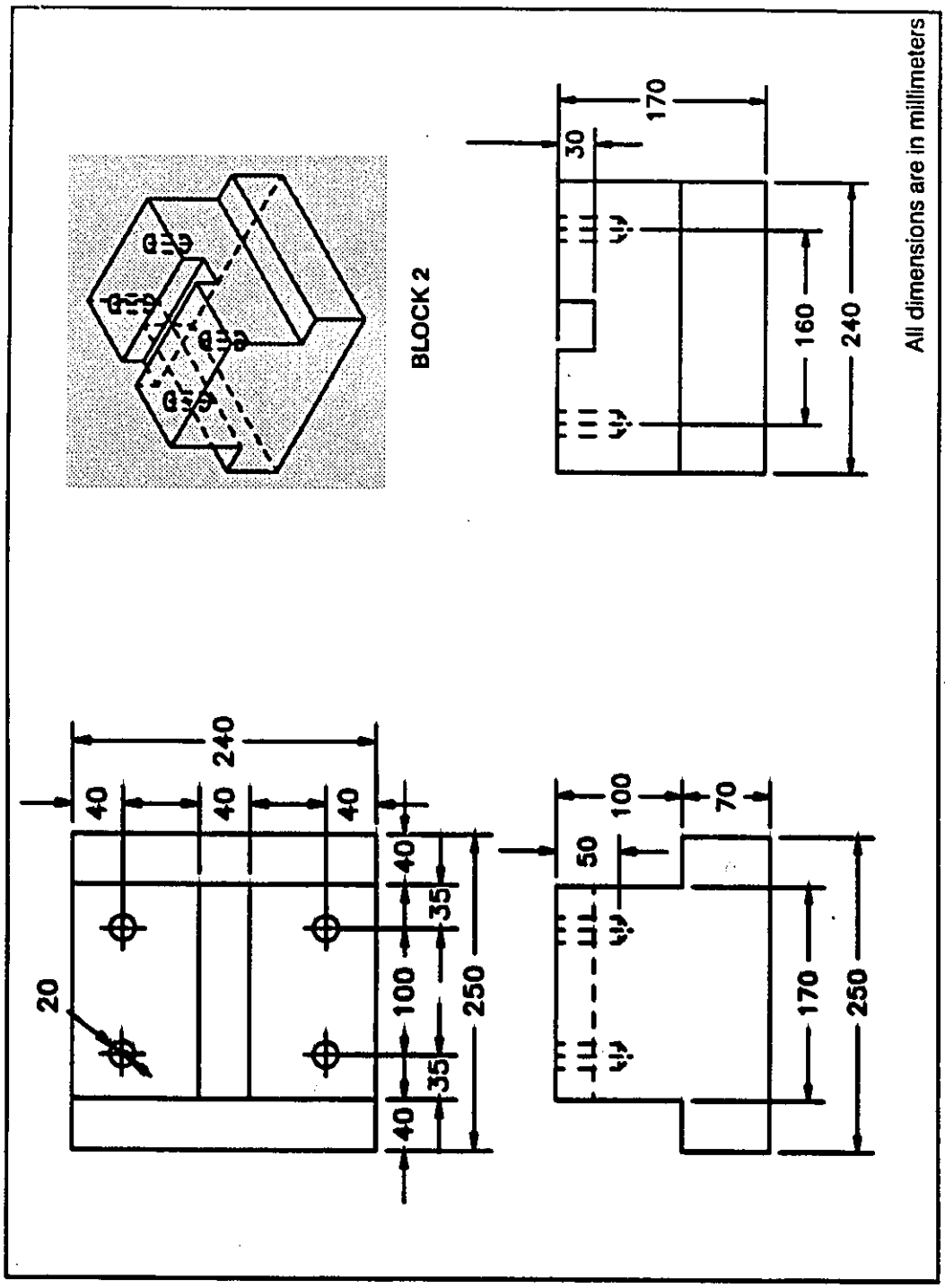
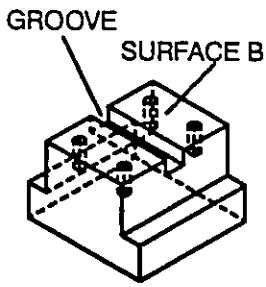


Figure 7.3. Drawing layout of BLOCK 2



PROCESS SEQUENCE	PROCESS CODE	SPECIAL TOOLS
Load block on pallet	LOAD−1	NO
Mill surface B Mill groove	MILL−2	NO
Drill four holes (Φ 20mm) on surface B	DRILL−2	NO
Unload block from pallet	UNLOAD−1	NO

BLOCK 2

Production volume − 150

Accuracy required − Low

Weight − 91 kgms

Figure 7.4. Process sequence of BLOCK 2

changes and the results from the knowledge−based design synthesizer are explained in the following sections.

Table 7.1. System requirements for case study 1

Length of floor:	**30 meters**
Breadth of floor:	**15 meters**
Expected utilization of workstations:	**70% − 80%**
Maximum blocking allowed at each workstation:	**10%**
Minimum production volume to achieve:	**98%**
Acceptable queue length / total queue length:	**75%**
Expected utilization of material handling systems:	**80% − 90%**
Planning horizon:	**3600 minutes**
Volume flexibility:	**High**
Expansion flexibility:	**High**
Routing flexibility:	**High**

Table 7.2. Components of initial FMS design for case study 1

Components	Component names
Pallets	PALLET.B.1, PALLET.B.2, PALLET.B.3, PALLET.C.1,
Workstations	LOAD.A.1, NCMM.A.1, NCMM.A.2, NCMM.A.3, UNLOAD.A.1
Material handling systems	AGV.B.1, AGV.B.2
Automatic pallet changers	APC.2.B.1, APC.2.B.2, APC.2.B.3, APC.2.B.4, APC.2.B.5

Table 7.3. Part type routing generated for case study 1

Part types	Routing
BLOCK 1	LOAD.A.1 –> (NCMM.A.1 NCMM.A.2 NCMM.A.3) –> UNLOAD.A.1
BLOCK 2	LOAD.A.1 –> (NCMM.A.1 NCMM.A.2 NCMM.A.3) –> UNLOAD.A.1

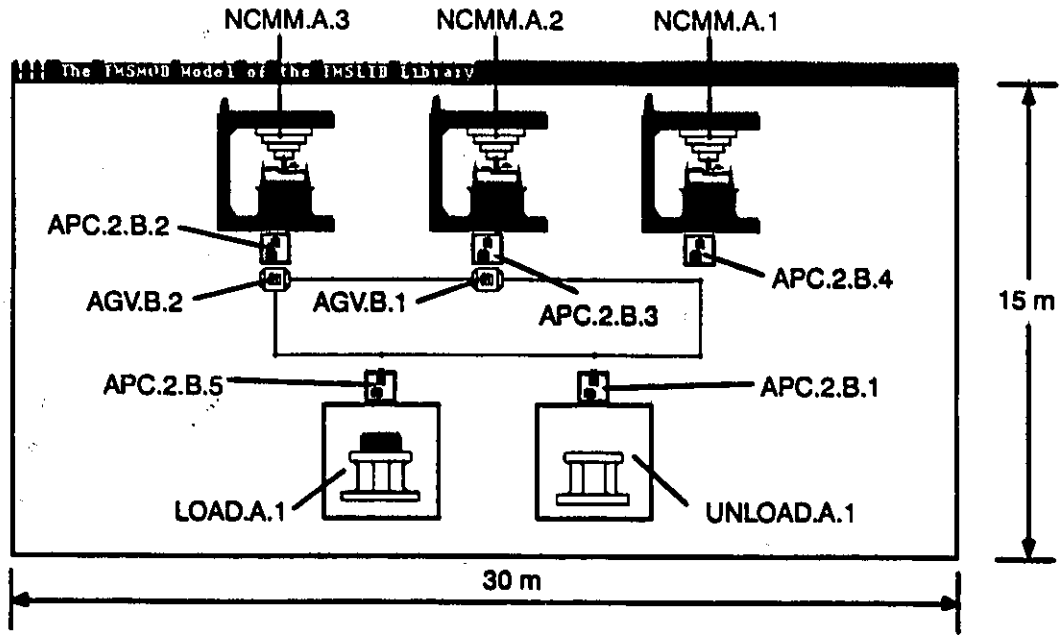


Figure 7.5. Graphical model of initial FMS design for case study 1

7.1.1.1 Input data variation – Case study 1A

In this case study, the accuracy required for both part types was changed from "low" to "high". The four holes to be drilled on each part type are to be done simultaneously using a special tool. The components of the initial FMS design developed by the design synthesizer are shown in Table 7.4.

Table 7.4. Components of initial FMS design for case study 1A

Components	Component names
Pallets	PALLET.B.1, PALLET.B.2, PALLET.B.3, PALLET.C.1
Workstations	LOAD.D.1, NCMM.D.1, NCMM.D.2, NCDM.D.1, UNLOAD.D.1
Material handling systems	AGV.B.1, AGV.B.2
Automatic pallet changers	APC.2.B.1, APC.2.B.2, APC.2.B.3, APC.2.B.4, APC.2.B.5

Because the dimensions of both part types were not changed, PALLET.B was selected for carrying BLOCK 1 and PALLET.C for BLOCK 2. Four types of workstations were selected to process this family of part types—LOAD.D, NCMM.D, NCDM.D, and UNLOAD.D—with high accuracy. Based on the production volume of both part types and the expected utilization of the workstations, the quantity of each workstation type was determined. One LOAD.D workstation, LOAD.D.1, one UNLOAD.D workstation, UNLOAD.D.1, two NCMM.D workstations, NCMM.D.1 and NCMM.D.2, and one NCDM.D workstation, NCDM.D.1, were required.

The routing for both part types produced by the design synthesizer are shown in Table 7.5. In this case, unlike the previous one, the second and third process in the process sequence for both part types were not combined by the design synthesizer into a single process. They are done by workstation type NCMM.D and NCDM.D, respectively. This is because of the special tools required for the third

process, DRILL−1 for BLOCK 1 and DRILL−2 for BLOCK 2. The second process for both part types could be performed by either NCMM.D.1 or NCMM.D.2.

Table 7.5. Part type routing generated for case study 1A

Part types	Routing
BLOCK 1	LOAD.D.1 −> (NCMM.D.1 NCMM.D.2) −> NCDM.1 −> UNLOAD.D.1
BLOCK 2	LOAD.D.1 −> (NCMM.D.1 NCMM.D.2) −> NCDM.1 −> UNLOAD.D.1

Three quantities of PALLET.B, PALLET.B.1, PALLET.B.2, and PALLET.B.3, and one of PALLET.C, PALLET.C.1, were required. Based on the number of pallets circulating in the system, each workstation is provided with an automatic pallet changer of capacity two. This is the minimum capacity required and five automatic pallet changers each with a capacity of two were selected as before—APC.2.B.1, APC.2.B.2, APC.2.B.3, APC.2.B.4, and APC.2.B.5.

Both part types visit the same workstation types because they have the same routing. Moreover, they do not require the loading station, LOAD.A.1, for either refixturing or reorienting since this is a serial flow situation. Based on this serial work flow and the type of system flexibilities (Table 7.1), automated guided vehicles (AGVs) were selected as the material handling system. Based on the maximum dimensions of the pallet type and combined weight of both part type and pallet type, AGV.B was selected and two vehicles were required—AGV.B.1 and AGV.B.2. Based on the type of material handling system (AGVs), the type of work flow, and the available floor space, a loop type layout was selected. The workstations were laid out on both sides of the loop to minimize the travel distance. The graphical simulation model of the design generated by the simulation model developer is shown in Figure 7.6.

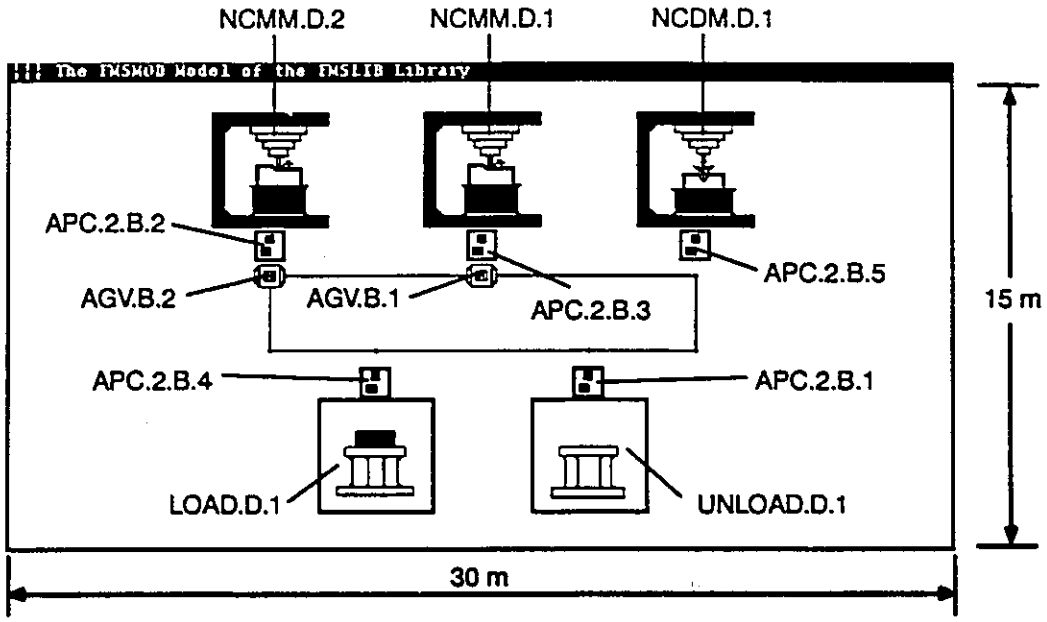


Figure 7.6. Graphical model of initial FMS design for case study 1A

7.1.1.2 Input data variation – Case study 1B

In this case study, the desired system flexibilities used in case study 1 were changed. The volume flexibility accuracy was changed to "low", the expansion flexibility was changed to "low", and the routing flexibility was changed to "medium". The components of the initial FMS design developed by the design synthesizer are shown in Table 7.6.

The only change in this design when compared to the original one is the type of material handling system used for transporting parts and pallets. In this case, conveyors were selected because of the change in system flexibilities. Seven conveyor segments, CONVEYOR.B.1 through CONVEYOR.B.7, were selected to complete the loop. The graphical simulation model of the design generated by the simulation model developer is shown in Figure 7.7.

Table 7.6. Components of initial FMS design for case study 1B

Components	Component names
Pallets	PALLET.B.1, PALLET.B.2, PALLET.B.3, PALLET.C.1,
Workstations	LOAD.A.1, NCMM.A.1, NCMM.A.2, NCMM.A.3, UNLOAD.A.1
Material handling systems	CONVEYOR.B.1, CONVEYOR.B.2, CONVEYOR.B.3, CONVEYOR.B.4, CONVEYOR.B.5, CONVEYOR.B.6, CONVEYOR.B.7
Automatic pallet changers	APC.2.B.1, APC.2.B.2, APC.2.B.3, APC.2.B.4, APC.2.B.5

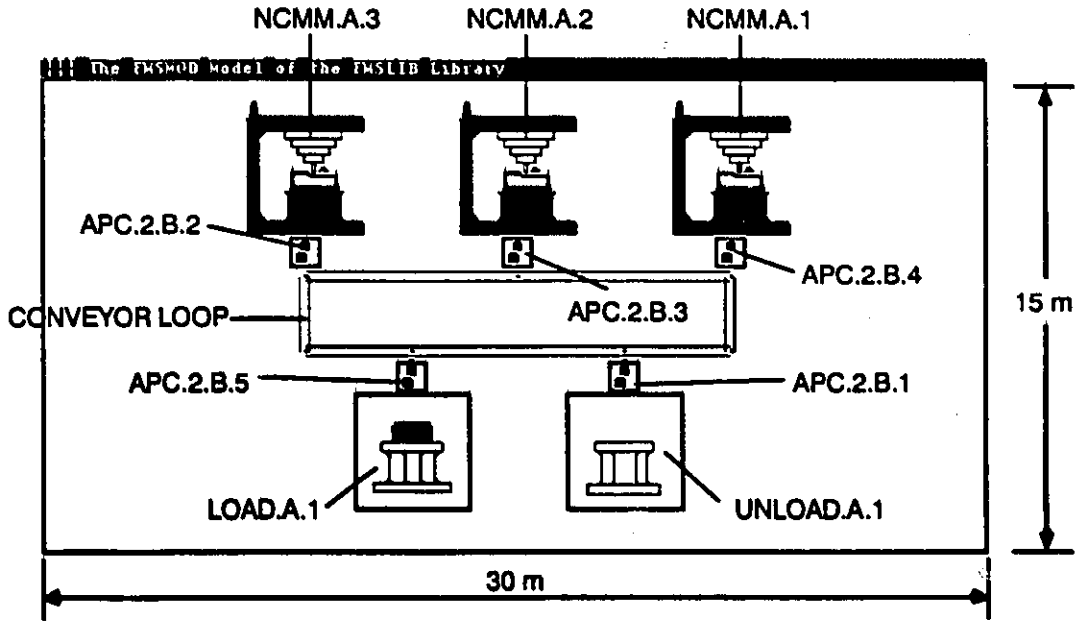


Figure 7.7. Graphical model of initial FMS design for case study 1B

7.1.1.3 Input data variation – Case study 1C

In this case study, the production volumes of both part types were changed—the production volume of BLOCK 1 was increased to 400 and that of

BLOCK 2 was increased to 200. The components of the FMS design developed by the design synthesizer are shown in Table 7.7.

Table 7.7. Components of initial FMS design for case study 1C

Components	Component names
Pallets	PALLET.B.1, PALLET.B.2, PALLET.B.3, PALLET.B.4, PALLET.C.1, PALLET.C.2
Workstations	LOAD.A.1, LOAD.A.2, NCMM.A.1, NCMM.A.2, NCMM.A.3, NCMM.A.4, UNLOAD.A.1, UNLOAD.A.2
Material handling systems	AGV.B.1, AGV.B.2, AGV.B.3, AGV.B.4
Automatic pallet changers	APC.2.B.1, APC.2.B.2, APC.2.B.3, APC.2.B.4, APC.2.B.5, APC.2.B.6, APC.2.B.7, APC.2.B.8

Based on the dimensions of both part types, PALLET.B was selected for carrying BLOCK 1 and PALLET.C for BLOCK 2. Three types of workstations were selected to process this family of part types—LOAD.A, NCMM.A, and UNLOAD.A. Based on the production volume of both part types and the expected utilization of the workstations, the quantity of each workstation type was determined. Two LOAD.A workstations, LOAD.A.1 and LOAD.A.2, two UNLOAD.A workstations, UNLOAD.A.1 and UNLOAD.A.2, and four NCMM.A workstations, NCMM.A.1, NCMM.A.2, NCMM.A.3, and NCMM.A.4, were required. The increase in the number of workstations of each type is a result of the increase in production volumes of both part types—more workstations are needed to meet these production volumes.

The routing for both part types produced by the design synthesizer are shown in Table 7.8. The second and third processes in the process sequence for both blocks, given their similarity, were combined by the design synthesizer into a single process which could be performed by either NCMM.A.1, NCMM.A.2, NCMM.A.3,

or NCMM.A.4. The loading process for both part types can be done by LOAD.A.1 or LOAD.A.2 while the unloading process can be done by UNLOAD.A.1 or UNLOAD.A.2.

Table 7.8. Part type routing generated for case study 1C

Part types	Routing
BLOCK 1	(LOAD.A.1 LOAD.A.2) –> (NCMM.A.1 NCMM.A.2 NCMM.A.3 NCMM.A.4) –> (UNLOAD.A.1 UNLOAD.A.2)
BLOCK 2	(LOAD.A.1 LOAD.A.2) –> (NCMM.A.1 NCMM.A.2 NCMM.A.3 NCMM.A.4) –> (UNLOAD.A.1 UNLOAD.A.2)

Four quantities of PALLET.B, PALLET.B.1, PALLET.B.2, PALLET.B.3, and PALLET.B.4, and two of PALLET.C, PALLET.C.1 and PALLET.C.2, were required. Based on the number of pallets circulating in the system, each workstation is provided with an automatic pallet changer of capacity two. This is the minimum capacity required and eight automatic pallet changers each with a capacity of two were selected—APC.2.B.1, APC.2.B.2, APC.2.B.3, APC.2.B.4, APC.2.B.5, APC.2.B.6, APC.2.B.7, and APC.2.B.8.

Both part types visit the same workstation types, i.e., they have the same routing. Moreover, they do not require the loading station, LOAD.A.1, for either refixturing or reorienting since this is a serial flow situation. Based on this work flow and the type of system flexibilities (Table 7.1), automated guided vehicles (AGVs) were selected as the material handling system. Based on the maximum dimensions of the pallet type and combined weight of both part type and pallet type, AGV.B was selected. Four vehicles were required—AGV.B.1, AGV.B.2, AGV.B.3, and AGV.B.4. Based on the type material handling system (AGVs), the type of work flow, and the available floor space, a loop type layout was selected. The

workstations were laid out on both sides of the loop to minimize the travel distance. Because of the increase in the number of workstations, the floor space occupied by the flexible manufacturing system is more than in case study 1. The graphical simulation model of this initial flexible manufacturing system design generated by the simulation model developer is shown in Figure 7.8.

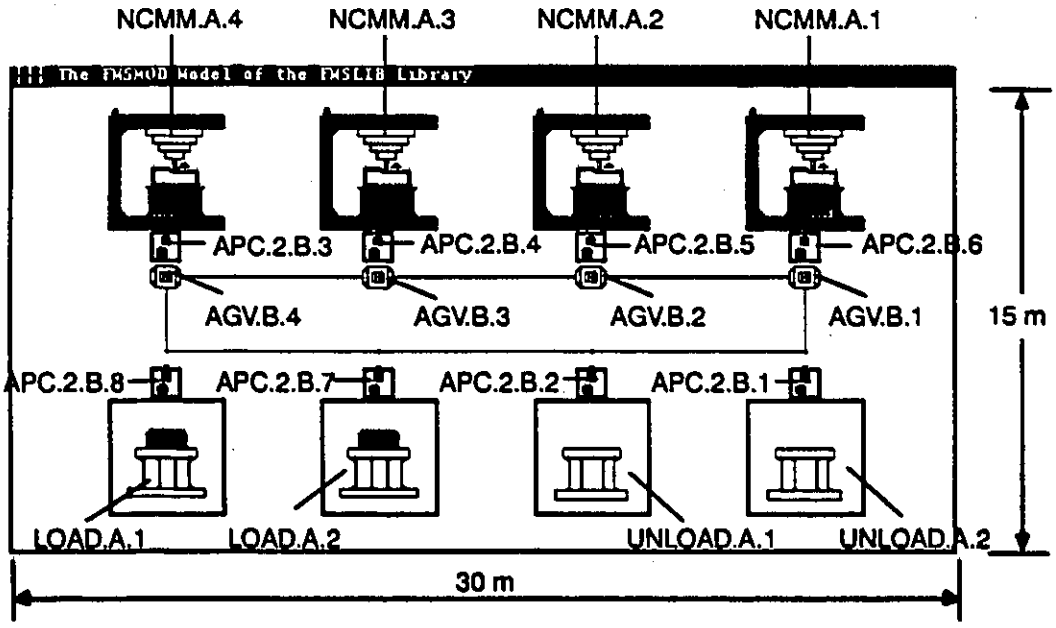


Figure 7.8. Graphical model of initial FMS design for case study 1C

7.1.1.4 Input data variation – Case study 1D

In this case study, the production volume of BLOCK 1 was decreased to 300 and two additional part types, BLOCK 3 and BLOCK 4 were added to the part family. BLOCK 3 is the same as BLOCK 1 except that the process sequence has an additional reaming operation, REAM–1, after DRILL–1. Similarly, BLOCK 4 is the same as BLOCK 2 except that the process sequence has an additional reaming operation, REAM–2, after DRILL–2. The batch size of BLOCK 3 is 40

while that of BLOCK 4 is 20. This input data was used to demonstrate FMX's ability to handle problems of a bigger size with more part types in a family. The components of the initial FMS design developed by the design synthesizer are shown in Table 7.9.

Table 7.9. Components of initial FMS design for case study 1D

Components	Component names
Pallets	PALLET.B.1, PALLET.B.2, PALLET.B.3, PALLET.B.4, PALLET.C.1, PALLET.C.2
Workstations	LOAD.A, NCMM.A.1, NCMM.A.2, NCMM.A.3, NCMM.A.4, UNLOAD.A.1
Material handling systems	AGV.B.1, AGV.B.2, AGV.B.3
Automatic pallet changers	APC.2.B.1, APC.2.B.2, APC.2.B.3, APC.2.B.4, APC.2.B.5, APC.2.B.6

Based on the dimensions of the four part types in the part family, PALLET.B was selected for carrying BLOCK 1 and BLOCK 3 while PALLET.C was selected for carrying BLOCK 2 and BLOCK 4. Three types of workstations were selected to process this family of part types—LOAD.A, NCMM.A, and UNLOAD.A. Based on the production volume of the part types and the expected utilization of the workstations, the number of each workstation type was determined. One LOAD.A workstations, LOAD.A.1, one UNLOAD.A workstations, UNLOAD.A.1, and four NCMM.A workstations, NCMM.A.1, NCMM.A.2, NCMM.A.3, and NCMM.A.4, were required. The increase in the number of workstations of each type is due to the increase in the number of part types—more workstations are needed to meet these production volumes in the same production time of 3600 minutes.

The routing for all four part types produced by the design synthesizer are

shown in Table 7.10. The second and third process in the process sequence for BLOCK 1 and BLOCK 2, given their simiiarity, were combined by the design synthesizer into a single process which could be performed by either NCMM.A.1, NCMM.A.2, NCMM.A.3, or NCMM.A.4. Simliarly, the second, third, and fourth process in the process sequence for BLOCK 3 and BLOCK 4 were combined into a single process which could be performed by either NCMM.A.1, NCMM.A.2, NCMM.A.3, or NCMM.A.4. The loading process for both part types is done by LOAD.A.1 while the unloading process is done by UNLOAD.A.1.

Table 7.10. Part type routing generated for case study 1D

Part types	Routing
BLOCK 1	(LOAD.A.1) -> (NCMM.A.1 NCMM.A.2 NCMM.A.3 NCMM.A.4) -> (UNLOAD.A.1)
BLOCK 2	(LOAD.A.1) -> (NCMM.A.1 NCMM.A.2 NCMM.A.3 NCMM.A.4) -> (UNLOAD.A.1)
BLOCK 3	(LOAD.A.1) -> (NCMM.A.1 NCMM.A.2 NCMM.A.3 NCMM.A.4) -> (UNLOAD.A.1)
BLOCK 4	(LOAD.A.1) -> (NCMM.A.1 NCMM.A.2 NCMM.A.3 NCMM.A.4) -> (UNLOAD.A.1)

Four quantities of PALLET.B, PALLET.B.1, PALLET.B.2, PALLET.B.3, and PALLET.B.4, and two of PALLET.C, PALLET.C.1, and PALLET.C.2, were required. Based on the number of pallets circulating in the system, each workstation is provided with an automatic pallet changer of capacity two. This is the minimum capacity required and six automatic pallet changers each with a capacity of two were selected—APC.2.B.1, APC.2.B.2, APC.2.B.3, APC.2.B.4, APC.2.B.5, and APC.2.B.6.

Based on the serial work flow and the type of system flexibilities

(Table 7.1), automated guided vehicles (AGVs) were selected as the material handling system. Based on the maximum dimensions of the pallet type and combined weight of both part type and pallet type, AGV.B was selected. Three vehicles were required—AGV.B.1, AGV.B.2, and AGV.B.3. Based on the type of material handling system (AGVs), the type of work flow, and the available floor space, a loop type layout was selected and workstations were laid out on both side of the loop to minimize the travel distance. Because of the increase in the number of workstations, the floor space utilized by this flexible manufacturing system is more than in case study 1. The graphical simulation model of this initial flexible manufacturing system design generated by the simulation model developer is shown in Figure 7.9.

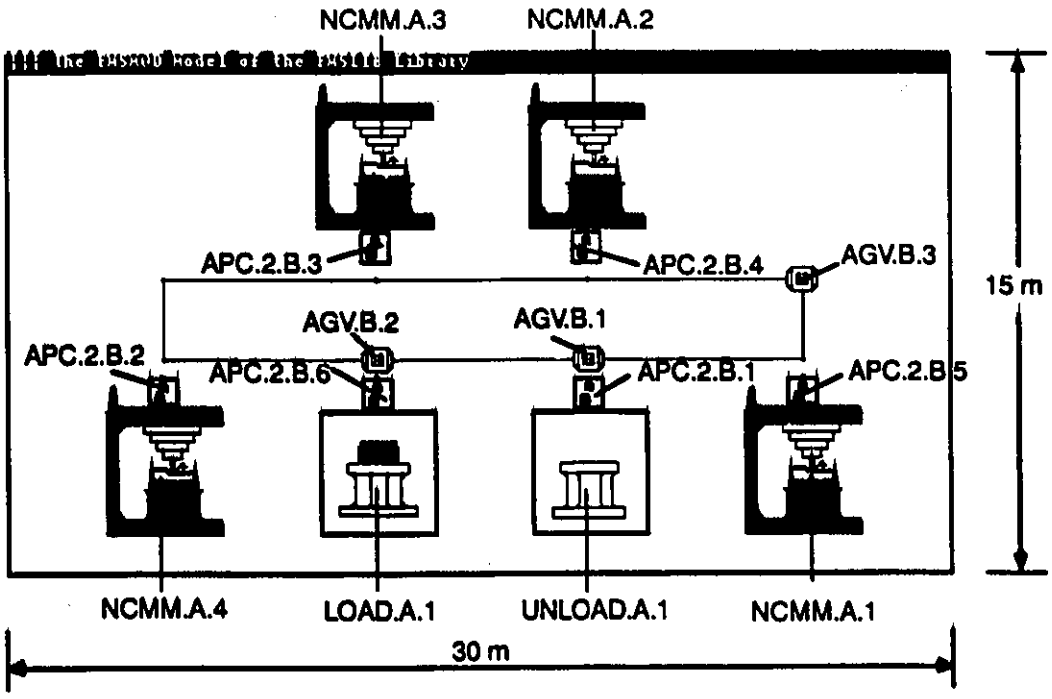


Figure 7.9. Graphical model of initial FMS design for case study 1D

7.1.2 Case study 2

In this industrial example, a new flexible manufacturing system is to be designed for machining two types of brake drums. The drawing layout of BRAKE DRUM 1 is shown Figure 7.10 and its process sequence is shown in Figure 7.11. The drawing layout of BRAKE DRUM 2 is shown in Figure 7.12 and its process sequence is shown in Figure 7.13. The system requirements are shown in Table 7.11. The components of the initial FMS design generated by the knowledge–based design synthesizer are shown in Table 7.12.

Based on the dimensions of the brake drums, PALLET.C was selected for carrying both. Seven types of workstations were selected to process this family of brake drums—LOAD.E, NCVTL.E NCDM.E, HONE.A, WASH.A, INSPECT.A, and UNLOAD.E. The required number of each workstation type was determined based on the production volumes of both brake drums and the expected utilization of the workstations. Except for the workstation type, NCVTL.E, the number of other workstation types were determined to be one. Three NCVTL.E workstations, NCVTL.E.1, NCVTL.E.2, and NCVTL.E.3, were required. The generated routing for the brake drums are shown in Table 7.13. The second, third, and fourth processes in the process sequence for both brake drums were combined by the design synthesizer into a single process which could be performed on NCVTL.E.1, NCVTL.E.2, or NCVTL.E.3.

Five PALLET.C types were required: PALLET.C.1, PALLET.C.2, and PALLET.C.3 for carrying BRAKE DRUM 1, and PALLET.C.4 and PALLET.C.5 for carrying BRAKE DRUM 2. Based on the number of pallets circulating in the system, each workstation is provided with an automatic pallet changer of capacity two. Therefore, nine automatic pallet changers with a capacity of two each were

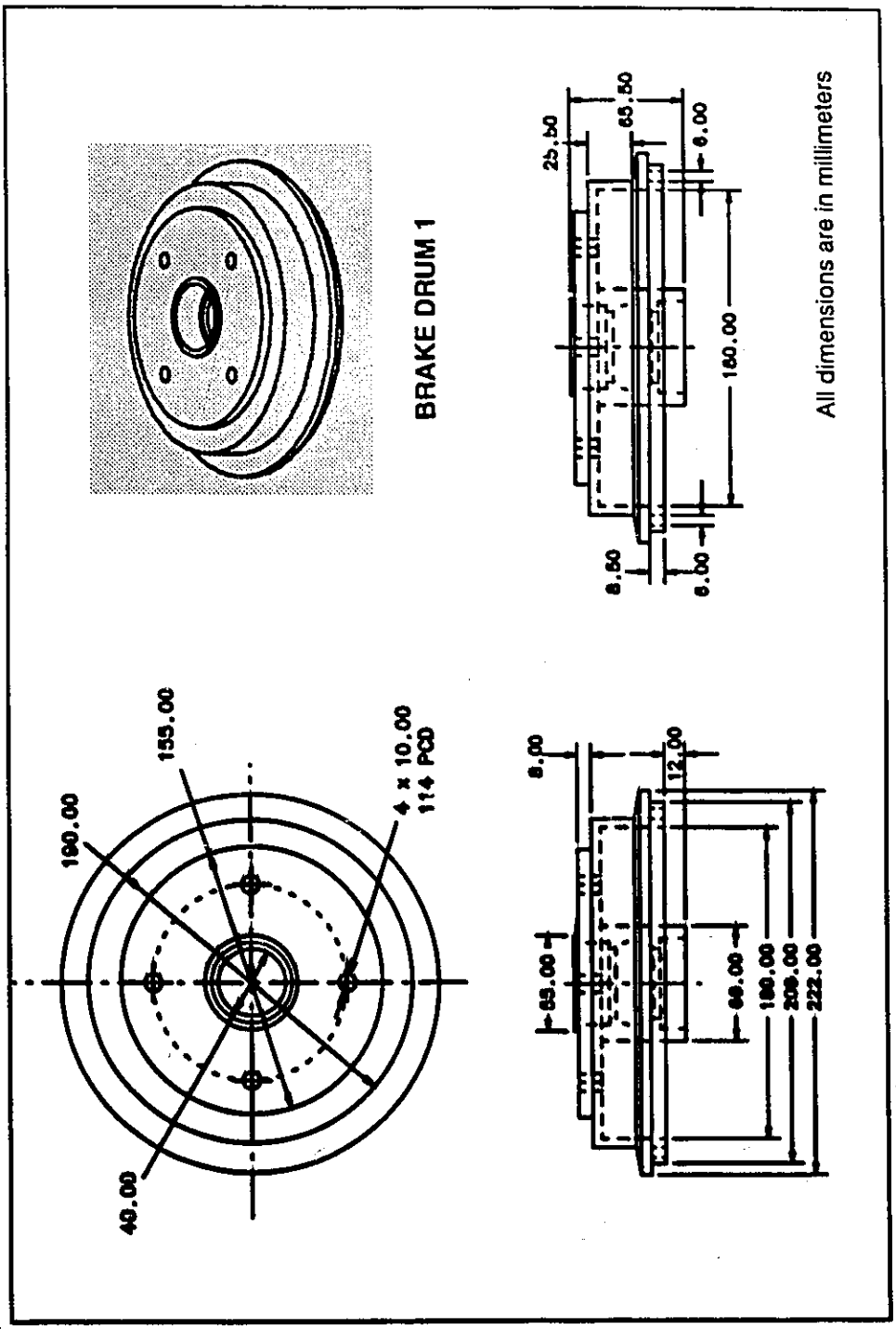
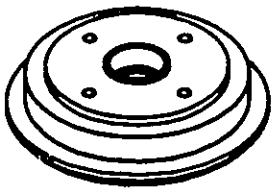


Figure 7.9. Drawing layout of BRAKE DRUM 1



BRAKE DRUM 1

PROCESS SEQUENCE	PROCESS CODE	SPECIAL TOOLS
Load brake drum on pallet	LOAD–2	NO
Turn outer surfaces	TURN–1	NO
Rough bore center hole	BORE–2	NO
Finish bore center hole	BORE–3	NO
Drill four holes (Φ 10mm) on brake drum	DRILL–3	YES
Hone center hole	HONE	NO
Wash brake drum	WASH	NO
Inspect brake drum	INSPECT–1	NO
Unload brake drum from pallet	UNLOAD–2	NO

Production volume – 350

Accuracy required – High

Weight – 8 kgms

Figure 7.11. Process sequence of BRAKE DRUM 1

selected—APC.2.C.1, APC.2.C.2, APC.2.C.3, APC.2.C.4, APC.2.C.5, APC.2.C.6, APC.2.C.7, APC.2.C.8, and APC.2.C.9.

Both drums visit the same workstation types, i.e., they have the same routing. Moreover, they do not visit the loading station, LOAD.A.1, for either refixturing or reorienting since this is a serial flow situation. Based on the type of work flow and the type of flexibilities (Table 7.11), conveyors were selected as the material handling system. Based on the maximum dimensions of the pallet types, CONVEYOR.C, was selected. Eleven conveyor segments, CONVEYOR.C.1 through CONVEYOR.C.11, were required. Based on the type of material handling system (conveyors), the type of work flow, and the available floor space, a loop type layout was selected. The workstations were laid out on both sides of the loop to

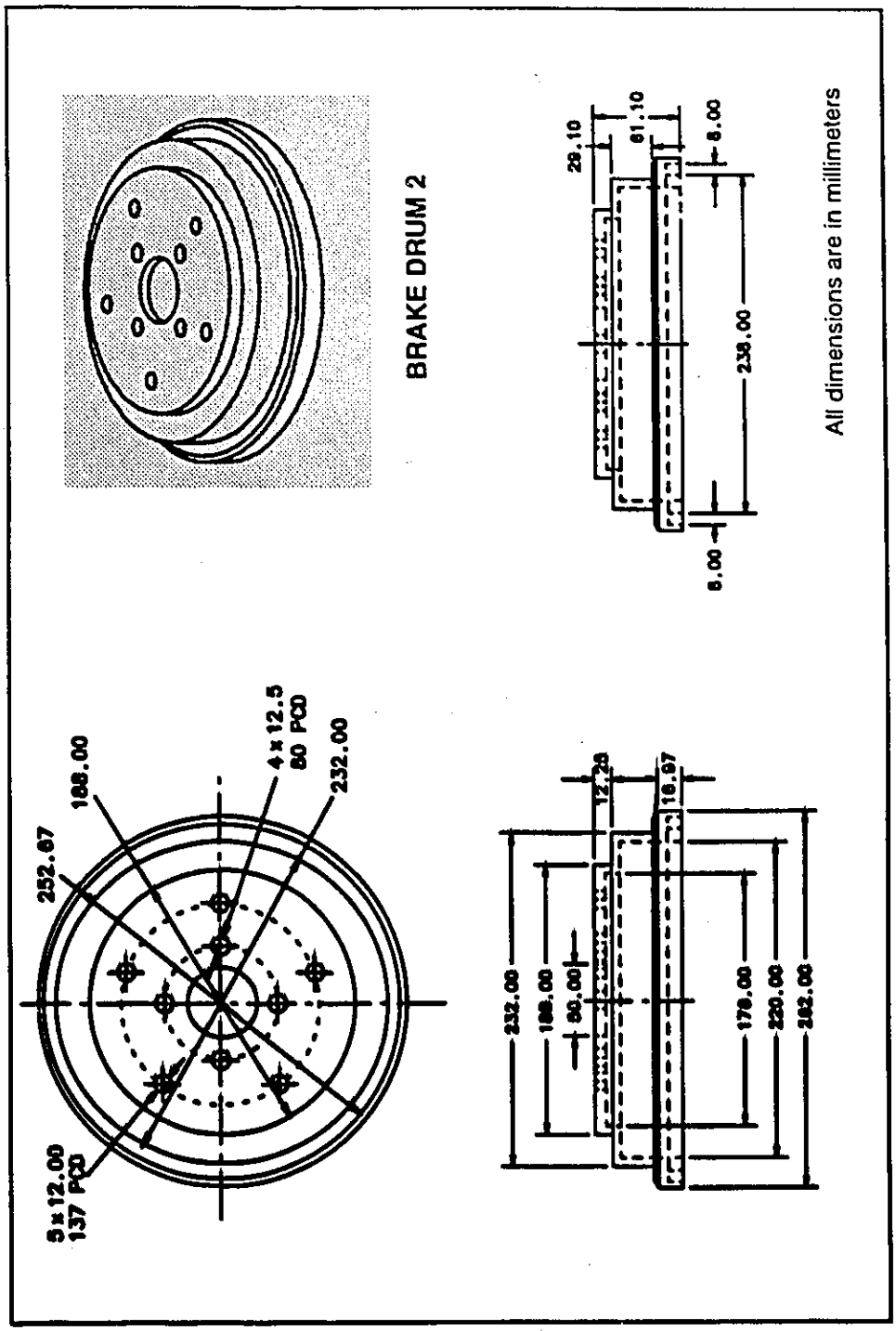
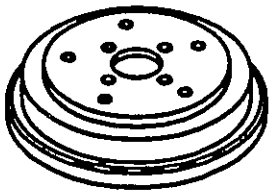


Figure 7.10. Drawing layout of BRAKE DRUM 2



BRAKE DRUM 2

PROCESS SEQUENCE	PROCESS CODE	SPECIAL TOOLS
Load brake drum on pallet	LOAD–2	NO
Turn outer surfaces	TURN–1	NO
Rough bore center hole	BORE–2	NO
Finish bore center hole	BORE–3	NO
Drill four holes (Φ 12.5mm) and five holes (Φ 12mm) on brake drum	DRILL–4	YES
Hone center hole	HONE	NO
Wash brake drum	WASH	NO
Inspect brake drum	INSPECT–1	NO
Unload brake drum from pallet	UNLOAD–2	NO

Production volume – 250

Accuracy required – High

Weight – 6 kgms

Figure 7.13. Process sequence of BRAKE DRUM 2

minimize the travel distance. The graphical simulation model of this initial flexible manufacturing system design generated by the simulation model developer is shown in Figure 7.14.

The actual manufacturing system that has been developed and installed for machining these brake drums has not been commissioned to full production yet and is still at a pilot stage. It was designed manually by the company engineers without the use of any simulation study. The design generated by the FMX design synthesizer module is different from the actual system in the following respects:

(i) The actual system layout is a U–shaped one with the loading station at one end of the U and the unloading stations at the other end and workstations on both sides of the U. The layout developed by the FMX design

synthesizer is a loop type layout with the load and unload stations next to each other in the loop. These two layouts are equivalent in terms of the unidirectional work flow and in both cases, parts are loaded into the system at the loading station and unloaded at the unloading station after they complete a loop through the system. The only difference is the physical connection between the unloading and the loading stations for transporting empty pallets back to the loading station.

(ii) The second, third, and fourth processes in the process sequence of the drums are performed by three different NC workstations in the actual system. In the design generated by the design synthesizer, these three processes are performed in one set-up on identical multi-purpose NC workstations. Three such multi-purpose workstations of this type were required to meet the production volumes of both brake drums. Therefore, in the actual system each brake drum has to visit each NC workstation sequentially while in the newly designed system it has to visit any one of the multi-purpose NC workstation that is available for processing. This increases the flexibility of the manufacturing system.

(iii) There is no local storage at the workstations in the actual system and the conveyor segments between workstations act as temporary storage units. The newly designed system, however, has an automatic pallet changer (local storage) with a capacity of two units at each workstation in addition to the temporary storage on the various conveyor segments.

Table 7.11. System requirements for case study 2

Length of floor:	30 meters
Breadth of floor:	15 meters
Expected utilization of workstations:	65% – 75%
Maximum blocking allowed at each workstation:	10%
Minimum production volume to achieve:	95%
Acceptable queue length / total queue length:	75%
Expected utilization of material handling systems:	65% – 75%
Planning horizon:	600 minutes
Volume flexibility:	Low
Expansion flexibility:	Low
Routing flexibility:	Medium

Table 7.12. Components of initial FMS design for case study 2

Components	Component names
Pallets	PALLET.C.1, PALLET.C.2, PALLET.C.3, PALLET.C.4, PALLET.C.5
Workstations	LOAD.E.1, NCVTL.E.1, NCVTL.E.2, NCVTL.E.3, NCDM.E.1, HONE.A.1, WASH.A.1, INSPECT.A.1, UNLOAD.E.1
Material handling systems	CONVEYOR.C.1, CONVEYOR.C.2, CONVEYOR.C.3, CONVEYOR.C.4, CONVEYOR.C.5, CONVEYOR.C.6, CONVEYOR.C.7, CONVEYOR.C.8, CONVEYOR.C.9, CONVEYOR.C.10, CONVEYOR.C.11
Automatic pallet changers	APC.2.C.1, APC.2.C.2, APC.2.C.3, APC.2.C.4, APC.2.C.5, APC.2.C.6, APC.2.C.7, APC.2.C.8, APC.2.C.9

Table 7.13. Part type routing generated for case study 2

Part types	Routing
BRAKE DRUM 1	LOAD.E.1 –> (NCVTL.E.1 NCVTL.E.2 NCVTL.E.3) –> NCDM.E.1 –> HONE.A.1 –> WASH.A.1 –> INSPECT.A.1 –> UNLOAD.E.1
BRAKE DRUM 2	LOAD.E.1 –> (NCVTL.E.1 NCVTL.E.2 NCVTL.E.3) –> NCDM.E.1 –> HONE.A.1 –> WASH.A.1 –> INSPECT.A.1 –> UNLOAD.E.1

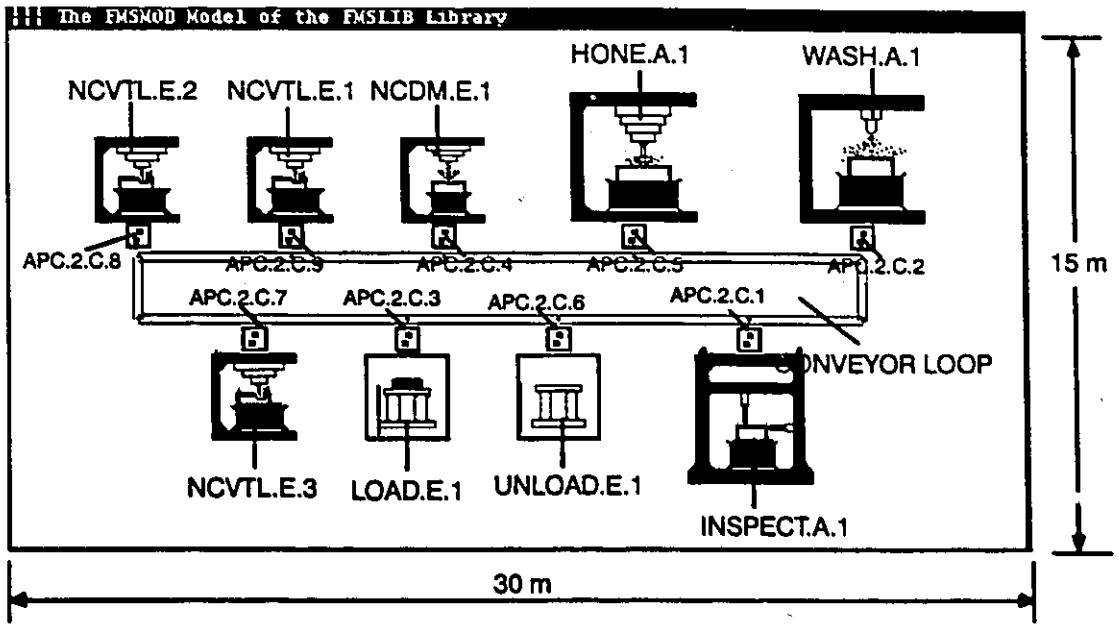


Figure 7.14. Graphical model of initial FMS design for case study 2

7.1.2.1 Input data variation – Case study 2A

This case study illustrates the response of the design synthesizer with respect to an increase in the number of part types to be processed. An additional brake drum, BRAKE DRUM 3, has to machined by the system. This new brake drum is similar to BRAKE DRUM 2 except that it does not require the center hole to be honed. The process sequence for BRAKE DRUM 3 is shown in Figure 7.15. The components of the initial FMS design generated by the design synthesizer are shown in Table 7.14.

Like the original case study, based on the dimensions of the brake drums, PALLET.C was selected for carrying both. Seven types of workstations were selected to process this family of brake drums—LOAD.E, NCVTL.E NCDM.E, HONE.A,

PROCESS SEQUENCE	PROCESS CODE	SPECIAL TOOLS
Load brake drum on pallet	LOAD–2	NO
Turn outer surfaces	TURN–1	NO
Rough bore center hole	BORE–2	NO
Finish bore center hole	BORE–3	NO
Drill four holes (Φ 12.5mm) and five holes (Φ 12mm) on brake drum	DRILL–4	YES
Wash brake drum	WASH	NO
Inspect brake drum	INSPECT–1	NO
Unload brake drum from pallet	UNLOAD–2	NO

Production volume – 300

Accuracy required – High

Weight – 6 kgms

Figure 7.15. Process sequence of BRAKE DRUM 3

WASH.A, INSPECT.A, and UNLOAD.E. The required number of each workstation type was determined based on the production volumes of three brake drums and the expected utilization of the workstations. Except for the workstation type, NCVTL.E and UNLOAD.E, the quantities of other workstation types were determined to be one. Four NCVTL.E workstations, NCVTL.E.1, NCVTL.E.2, NCVTL.E.3, and NCVTL.E.4 were required. Also two UNLOAD.E workstations, UNLOAD.E.1 AND UNLOAD.E.2, were required as compared to the one LOAD.E workstation. This is due to the increase in the overall production volume by the addition of the new brake drum and a longer unloading time than the loading time.

The generated routing for the brake drums are shown in Table 7.15. The

second, third, and fourth processes in the process sequence for all three brake drums were combined by the design synthesizer into a single process which could be performed on NCVTL.E.1, NCVTL.E.2, NCVTL.E.3, or NCVTL.E.4. Also, because BRAKE DRUM 3 has no honing operation, it does not visit HONE.A.1.

Seven PALLET.C types were required: PALLET.C.1, PALLET.C.2, and PALLET.C.3 for carrying BRAKE DRUM 1; PALLET.C.4 and PALLET.C.5 for carrying BRAKE DRUM 2; and PALLET.C.6 and PALLET.C.7 for carrying BRAKE DRUM 3. Based on the number of pallets circulating in the system, each workstation is provided with an automatic pallet changer of capacity two. Therefore, eleven automatic pallet changers with a capacity of two each were selected—APC.2.C.1, APC.2.C.2, APC.2.C.3, APC.2.C.4, APC.2.C.5, APC.2.C.6, APC.2.C.7, APC.2.C.8, APC.2.C.9, APC.2.C.10, and APC.2.C.11.

Conveyors were selected as the material handling system to transport palletized parts between the workstations. Based on the maximum dimensions of the pallet types CONVEYOR.C was selected. Thirteen conveyor segments, CONVEYOR.C.1 through CONVEYOR.C.13, were required. A loop type layout was selected and the workstations were laid out on both sides of the loop to minimize the travel distance. The graphical simulation model of this initial flexible manufacturing system design generated by the simulation model developer is shown in Figure 7.16.

Table 7.14. Components of initial FMS design for case study 2A

Components	Component names
Pallets	PALLET.C.1, PALLET.C.2, PALLET.C.3, PALLET.C.4, PALLET.C.5, PALLET.C.6, PALLET.C.7
Workstations	LOAD.E.1, NCVTL.E.1, NCVTL.E.2, NCVTL.E.3, NCVTL.E.4, NCDM.E.1, HONE.A.1, WASH.A.1, INSPECT.A.1, UNLOAD.E.1, UNLOAD.E.2
Material handling systems	CONVEYOR.C.1, CONVEYOR.C.2, CONVEYOR.C.3, CONVEYOR.C.4, CONVEYOR.C.5, CONVEYOR.C.6, CONVEYOR.C.7, CONVEYOR.C.8, CONVEYOR.C.9, CONVEYOR.C.10, CONVEYOR.C.11, CONVEYOR.C.12, CONVEYOR.C.13
Automatic pallet changers	APC.2.C.1, APC.2.C.2, APC.2.C.3, APC.2.C.4, APC.2.C.5, APC.2.C.6, APC.2.C.7, APC.2.C.8, APC.2.C.9, APC.2.C.10, APC.2.C.11

Table 7.15. Part type routing generated for case study 2A

Part types	Routing
BRAKE DRUM 1	LOAD.E.1 –> (NCVTL.E.1 NCVTL.E.2 NCVTL.E.3 NCVTL.E.4) –> NCDM.E.1 –> HONE.A.1 –> WASH.A.1 –> INSPECT.A.1 –> (UNLOAD.E.1 UNLOAD.E.2)
BRAKE DRUM 2	LOAD.E.1 –> (NCVTL.E.1 NCVTL.E.2 NCVTL.E.3 NCVTL.E.4) –> NCDM.E.1 –> HONE.A.1 –> WASH.A.1 –> INSPECT.A.1 –> (UNLOAD.E.1 UNLOAD.E.2)
BRAKE DRUM 3	LOAD.E.1 –> (NCVTL.E.1 NCVTL.E.2 NCVTL.E.3 NCVTL.E.4) –> NCDM.E.1 –> WASH.A.1 –> INSPECT.A.1 –> (UNLOAD.E.1 UNLOAD.E.2)

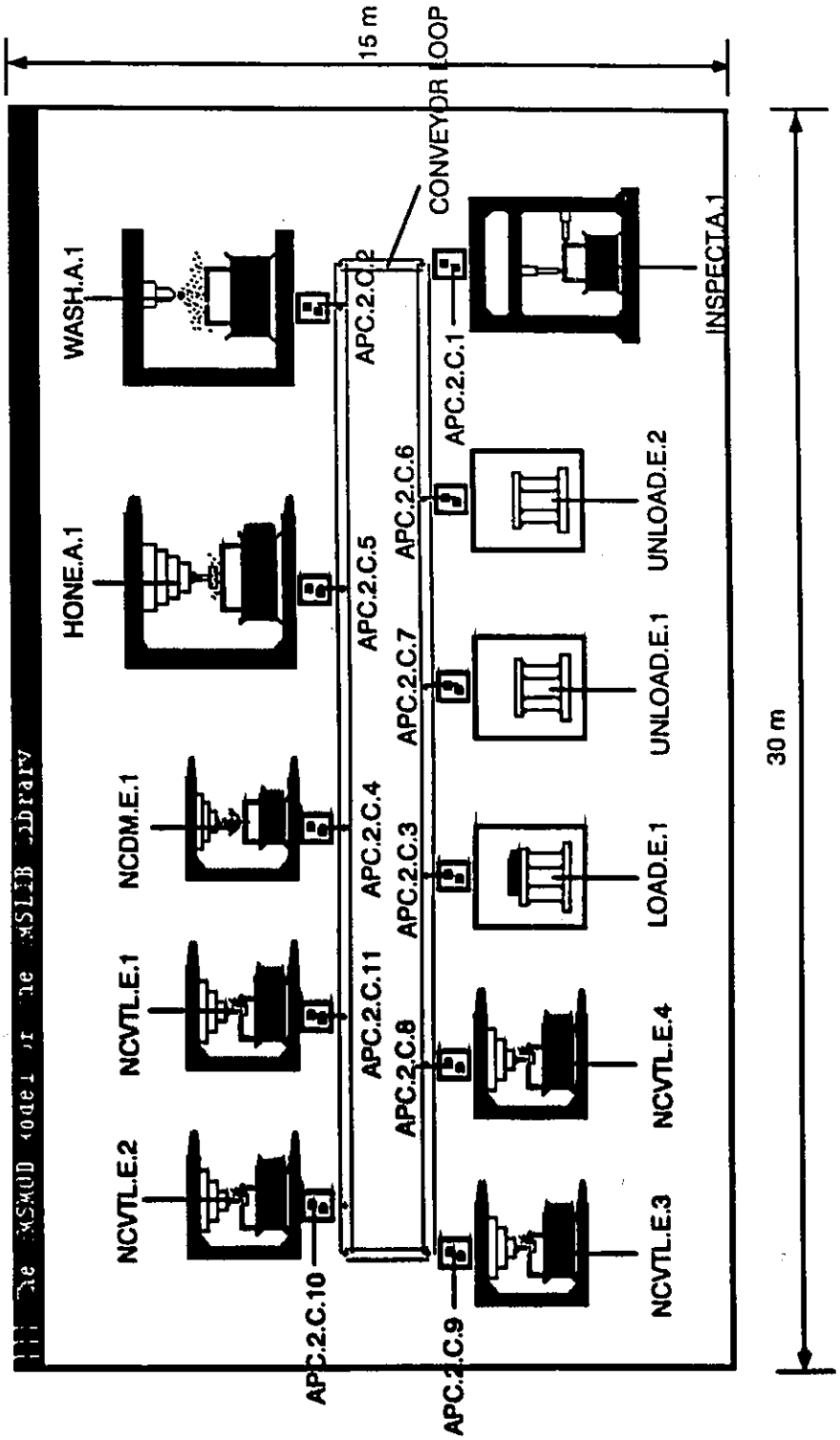


Figure 7.16. Graphical model of initial FMS design for case study 2A

7.2 Case study 1 – Fine tuning original design

The performance of the FMS design developed in case study 1 (Figure 7.5) was then evaluated by simulating the model and the data collected during the simulation was analyzed by the analyzer. This procedure of developing the simulation model, simulating the model, and analyzing the results was repeated automatically by FMX. Two iterations were required for this case study. In the first iteration, the speed of AGV.B.1, and AGV.B.2 was increased to 4 meters/minute. In the second iteration, the quantity of PALLET.B was increased by 2 and that of PALLET.C was increased by 1. These changes made to the design during each iteration are listed in Table 7.16.

The production volumes of BLOCK 1 and BLOCK 2 to be achieved within a time of 3600 minutes are 340 and 150, respectively. However, an acceptable production volume of 98% was set to be achieved for both part types—333 for BLOCK 1 and 147 for BLOCK 2.

Table 7.16. Design changes during each iteration for case study 1

Iteration	Changes made
1	Increase speed of AGV.B.1 to 4 meters/minute Increase speed of AGV.B.2 to 4 meters/minute
2	Increase PALLET.B by 2 (PALLET.B.4 and PALLET.B.5) Increase PALLET.C by 1 (PALLET.C.2)

The production volumes of both part types (BLOCK 1 and BLOCK 2) for each iteration are shown in Figure 7.17. The expected utilization of the each workstation is between 70% and 80%. The average utilization of each workstation for each iteration is shown collectively in Figure 7.18. Plots of utilization versus the

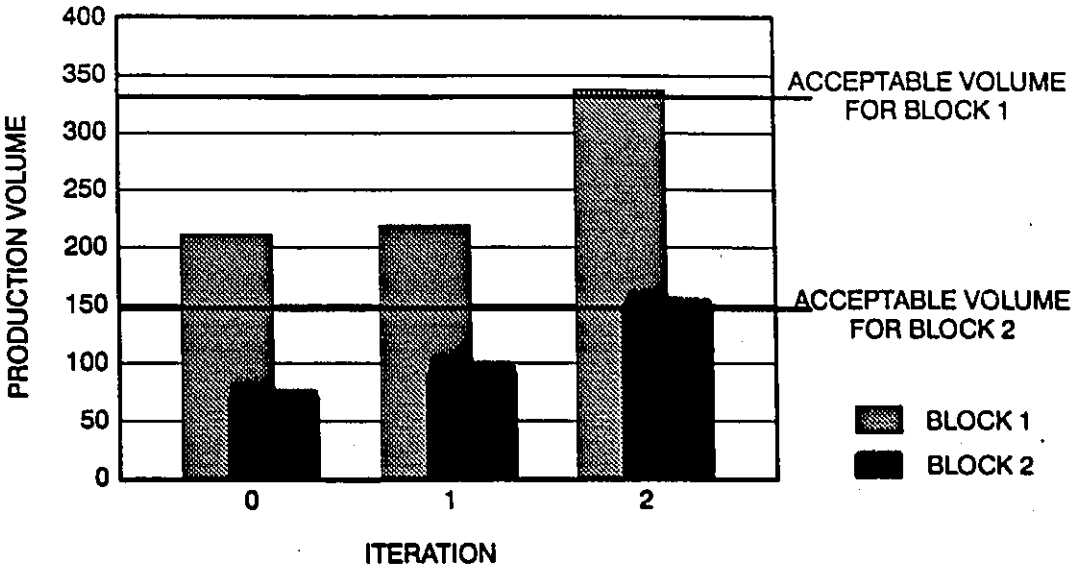


Figure 7.17. Production volumes of part types

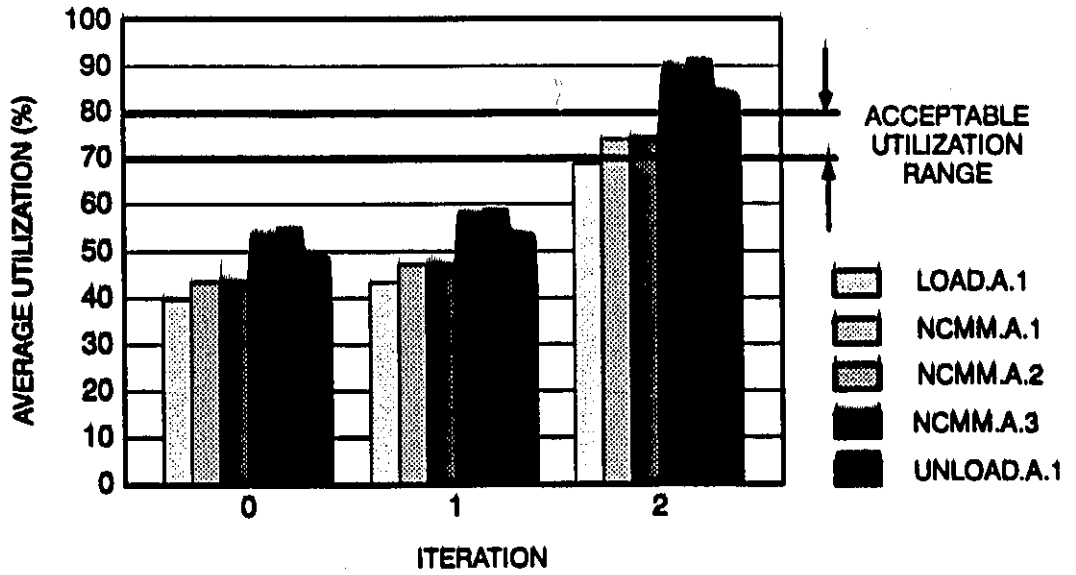


Figure 7.18. Average utilization of workstations

time of each workstation for each iteration are shown in Figure 7.19, Figure 7.20,

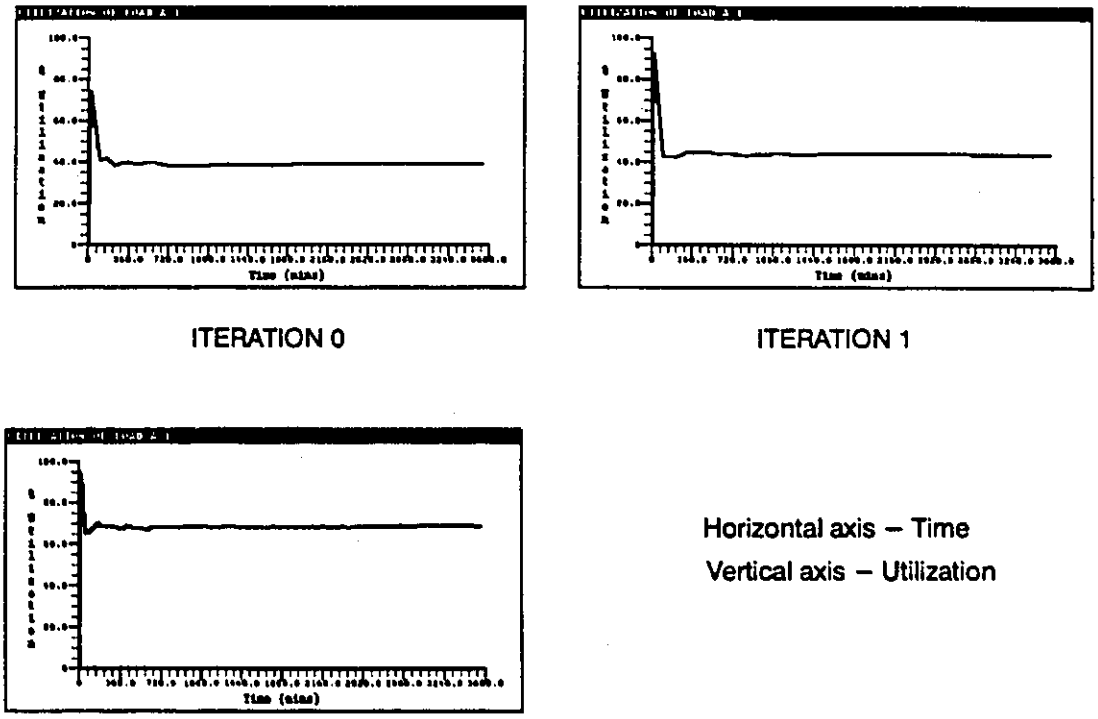


Figure 7.19. Utilization versus time of LOAD.A.1

Figure 7.21, Figure 7.22, and Figure 7.23. Individual plots of average utilization of each workstation for each iteration are shown in Figure 7.24, Figure 7.25, Figure 7.26, Figure 7.27, and Figure 7.28.

The maximum percentage of time each workstation could be blocked is 10%. The percentage of time each workstation was blocked for each iteration is shown in Figure 7.29.

The expected utilization of each AGV is between 80% and 90%. The average utilization of each AGV for each iteration is shown collectively in

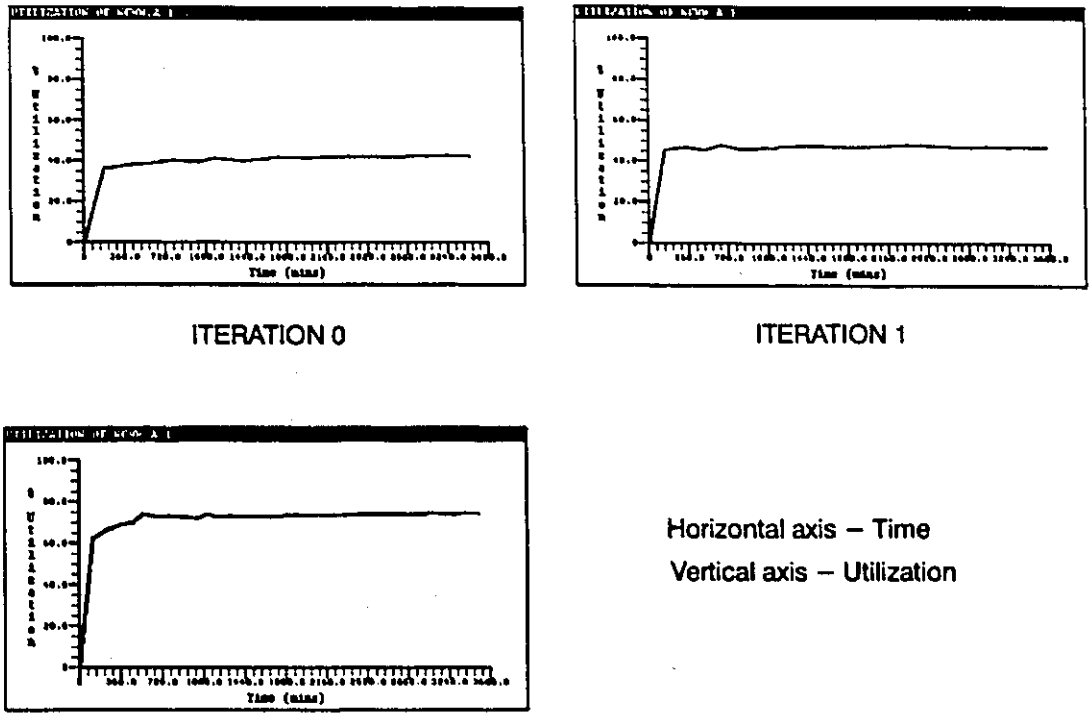


Figure 7.20. Utilization versus time of NCMM.A.1

Figure 7.30. Plots of utilization versus the time of each AGV for each iteration are shown in Figure 7.31 and Figure 7.32. Individual plots of average utilization of each AGV for each iteration are shown in Figure 7.33 and Figure 7.34.

7.2.1 Iteration 0

Iteration 0 corresponds to the initial design developed by the design synthesizer. After this design was simulated and the data collected, it was found that both part types did not meet their acceptable production volumes (Figure 7.17). All the workstations were under–utilized (Figure 7.18) and so were the AGVs

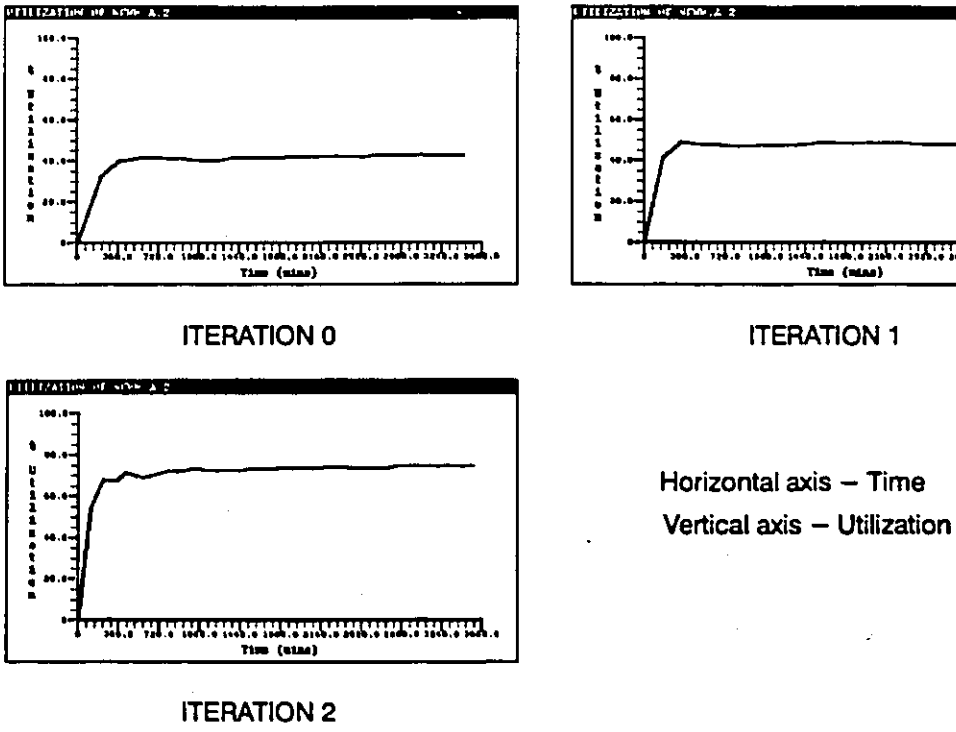


Figure 7.21. Utilization versus time of NCMM.A.2

(Figure 7.30). The blocking of the workstations was below the 10% preset value (Figure 7.29). Of the three possible recommendations to increase system throughput, namely, increasing the speed of both AGVs, increasing the quantity of AGV.B, and increasing the quantity of PALLET.B and PALLET.C, the analyzer chose to increase the speed of AGV.B.1 and AGV.B.2 because this change does not require any additional investment. Moreover, increasing the quantity of AGVs when both of them are under–utilized would only decrease their utilization further.

The new speed was calculated based on the current quantity of AGVs, the current production volume of both part types, and the expected production volume. This was determined to be 4 meters/minute which is the maximum travel speed for

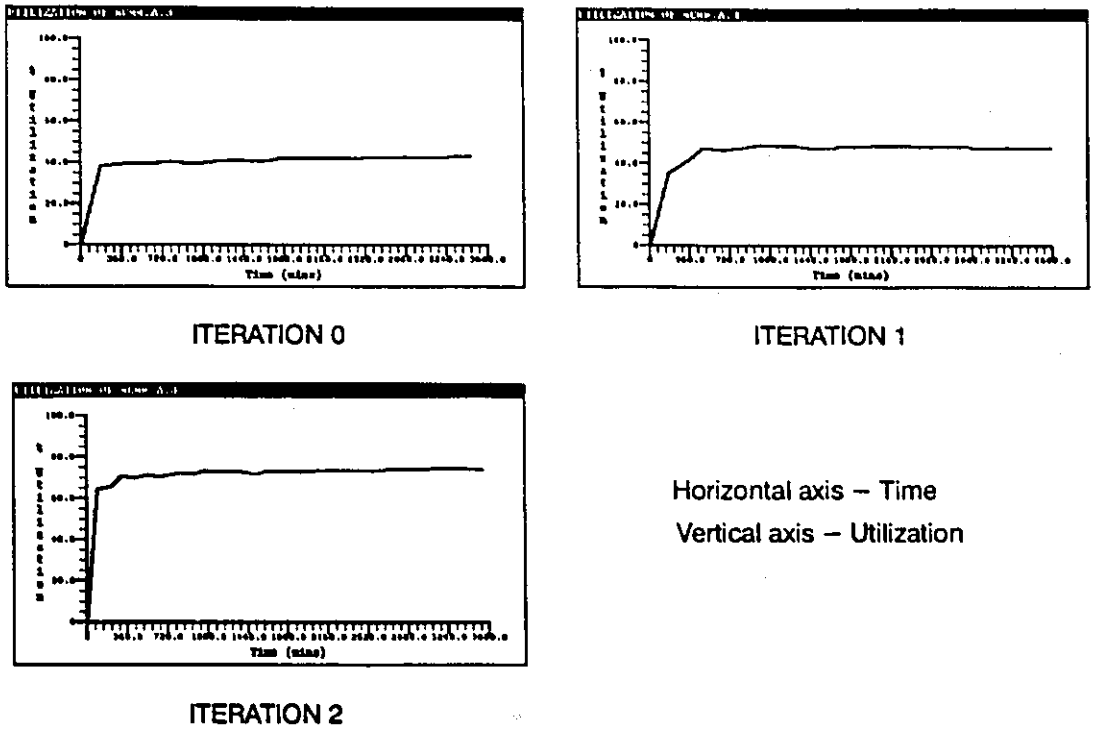


Figure 7.22. Utilization versus time of NCMM.A.3

AGV.B. Therefore, the new design was the same as the earlier on but with an increased travel speed for both AGVs—AGV.B.1 and AGV.B.2.

7.2.2 Iteration 1

In iteration 1, there is a small increase in production volume of both part types (Figure 7.17). The average utilization of the workstations increased only by a small amount and they were still lower than 70% (Figure 7.18). The blocking of the workstations was insignificant again (Figure 7.29). However, the utilization of AGV.B.1 and AGV.B.2 decreased (Figure 7.30). This is due to the increased travel speed which reduces the effective travel time of the AGVs and results in a lower utilization.

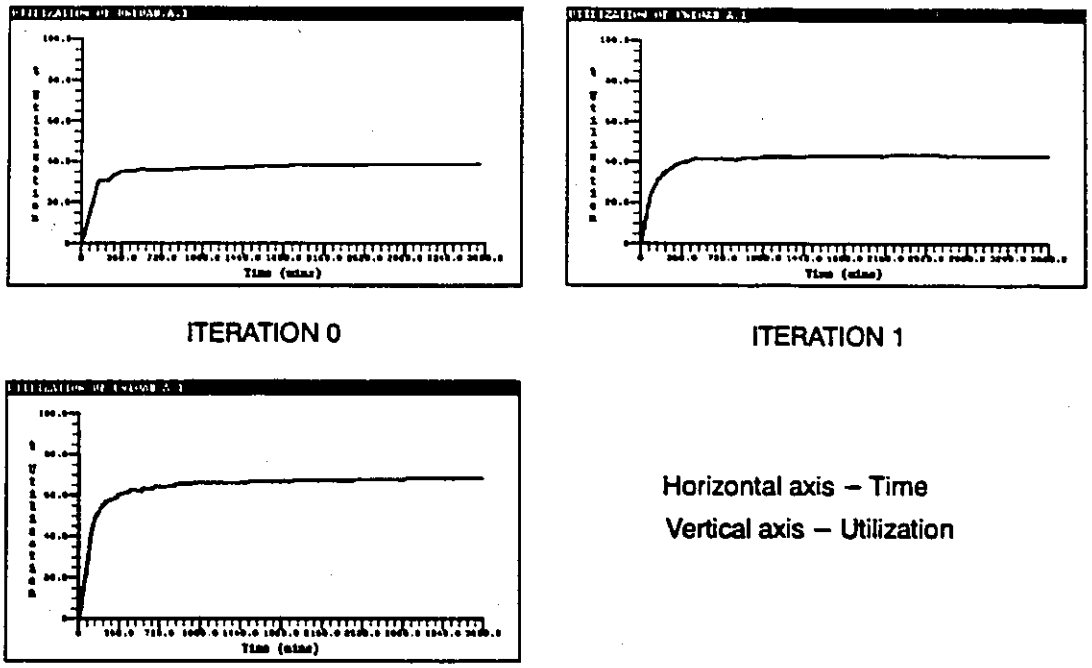


Figure 7.23. Utilization versus time of UNLOAD.A.1

In this situation, the analyzer again has three recommendations to increase the system throughput, namely, increasing the speed of the AGVs, increasing their quantity, or increasing the number of pallets. The speed of AGV.B.1 and AGV.B2 cannot be increased any further as they are already traveling at maximum speed. and the quantity of AGV.B cannot be increased because both AGVs are already under–utilized. The analyzer chose to increase the number of pallets of both part types.

The quantity by which these pallet types are be increased are based upon the current number, the current production volume of the part type being carried, and the expected production volume of this part type. The quantity of PALLET.B was increased by 2 while that of PALLET.C was increased by 1. The existing

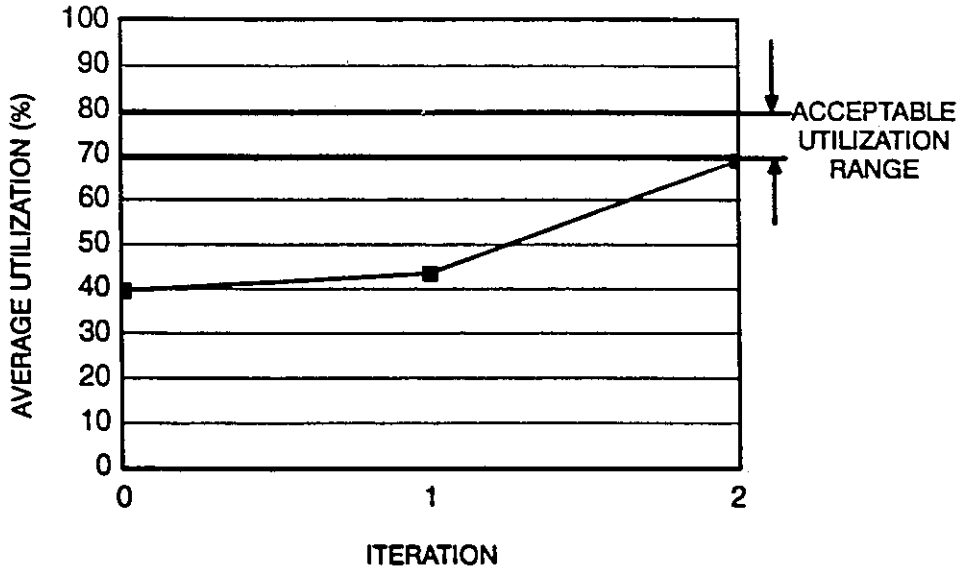


Figure 7.24. Average utilization of LOAD.A.1

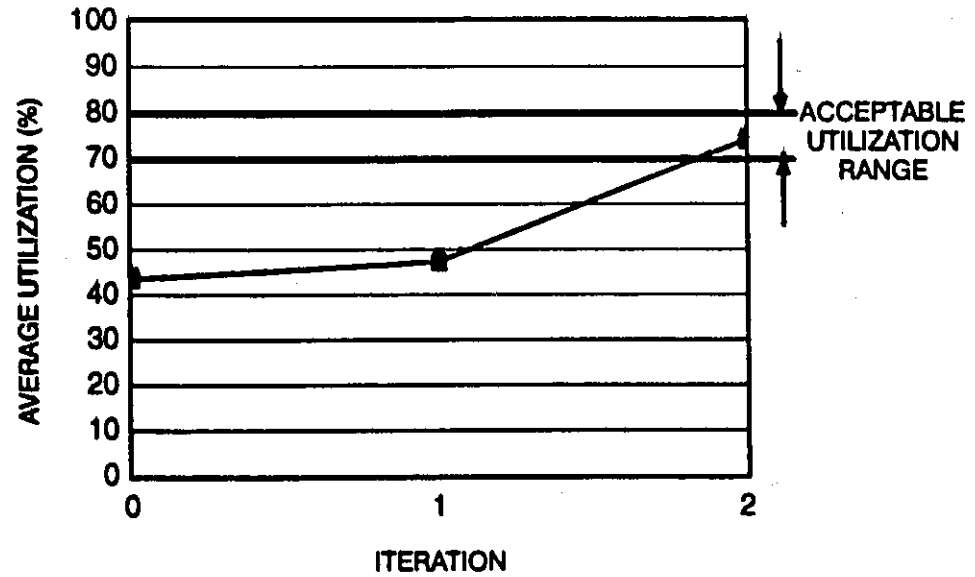


Figure 7.25. Average utilization of NCMM.A.1

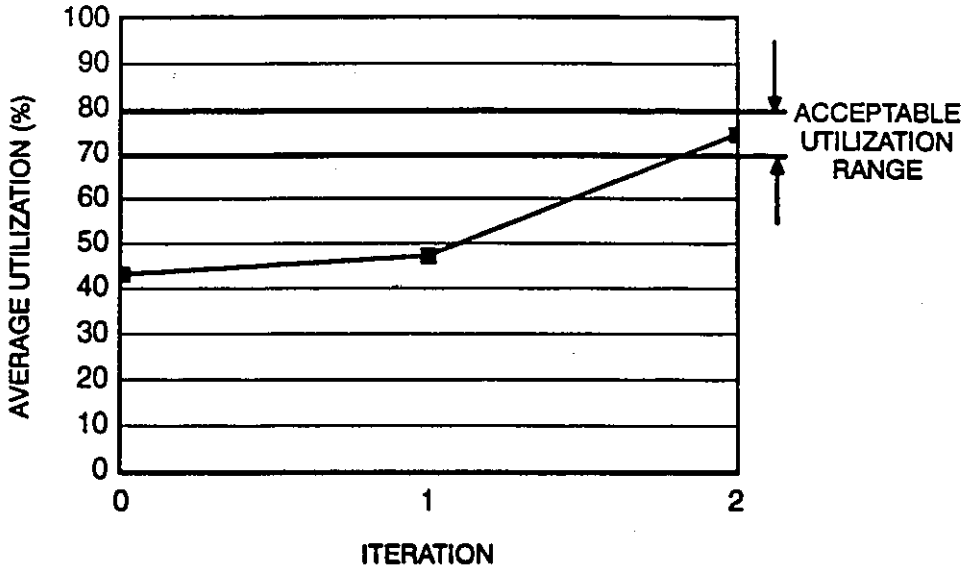


Figure 7.26. Average utilization of NCMM.A.2

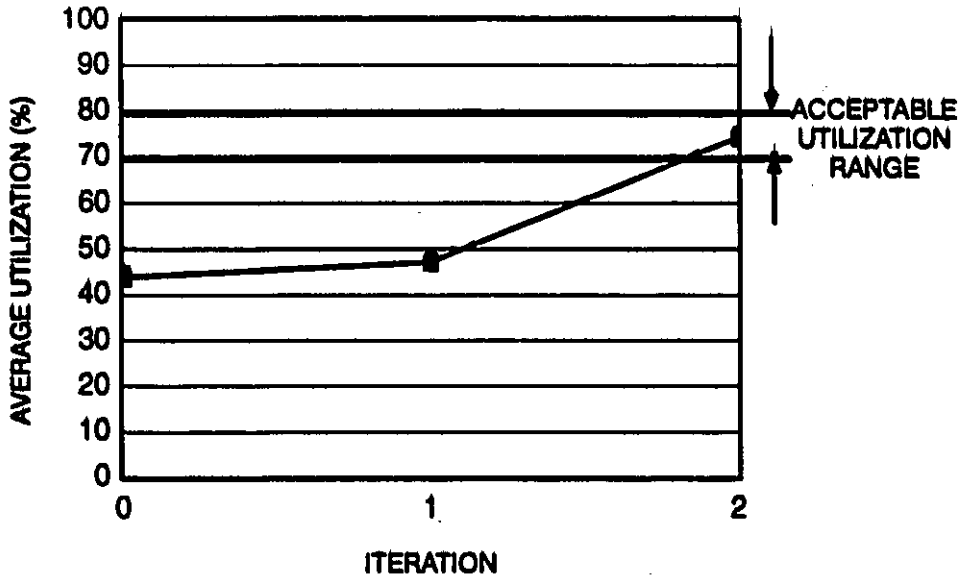


Figure 7.27. Average utilization of NCMM.A.3

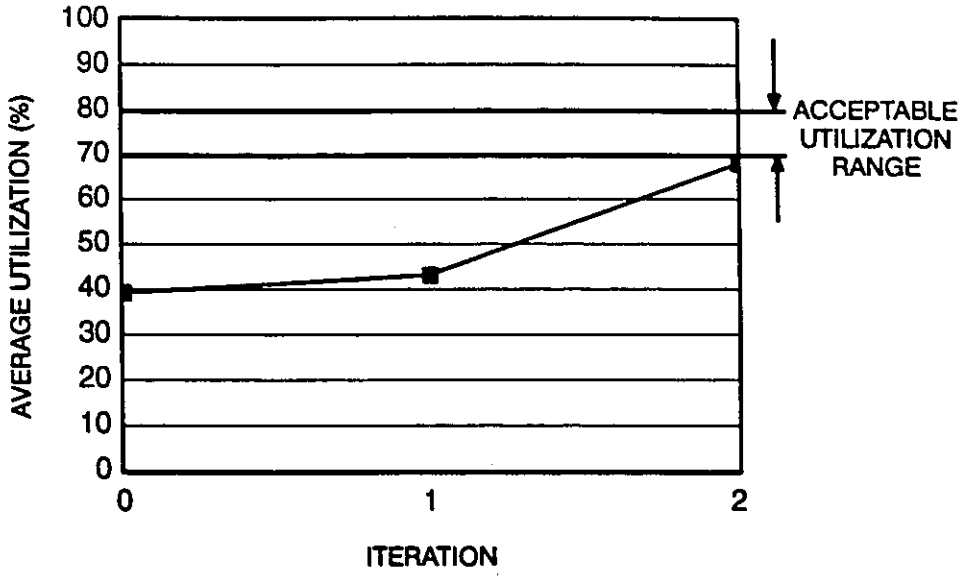


Figure 7.28. Average utilization of UNLOAD.A.1

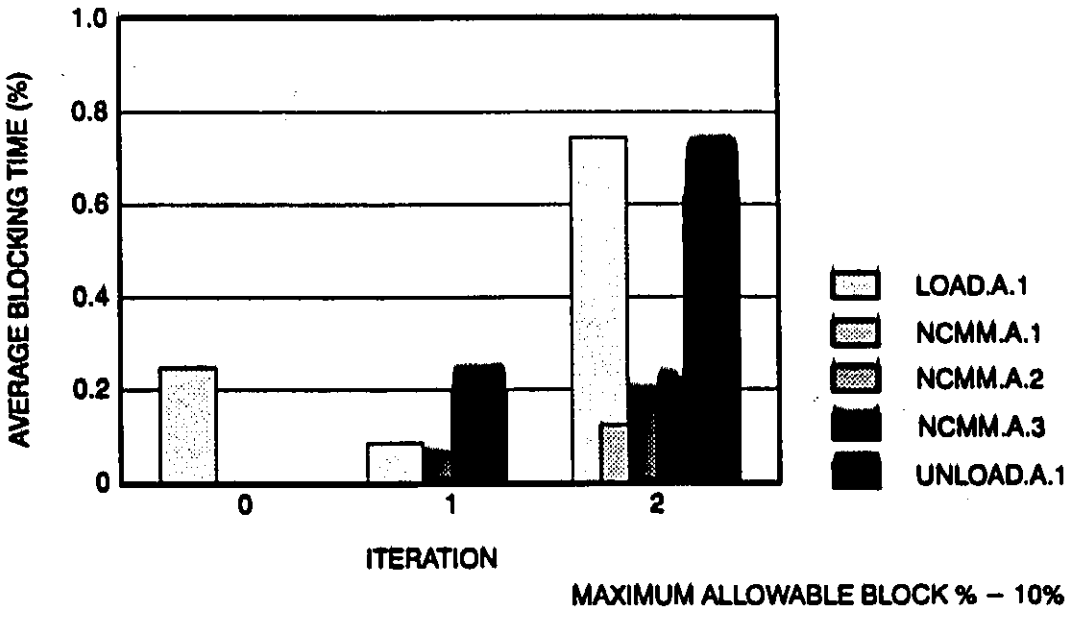


Figure 7.29. Blocking of workstations

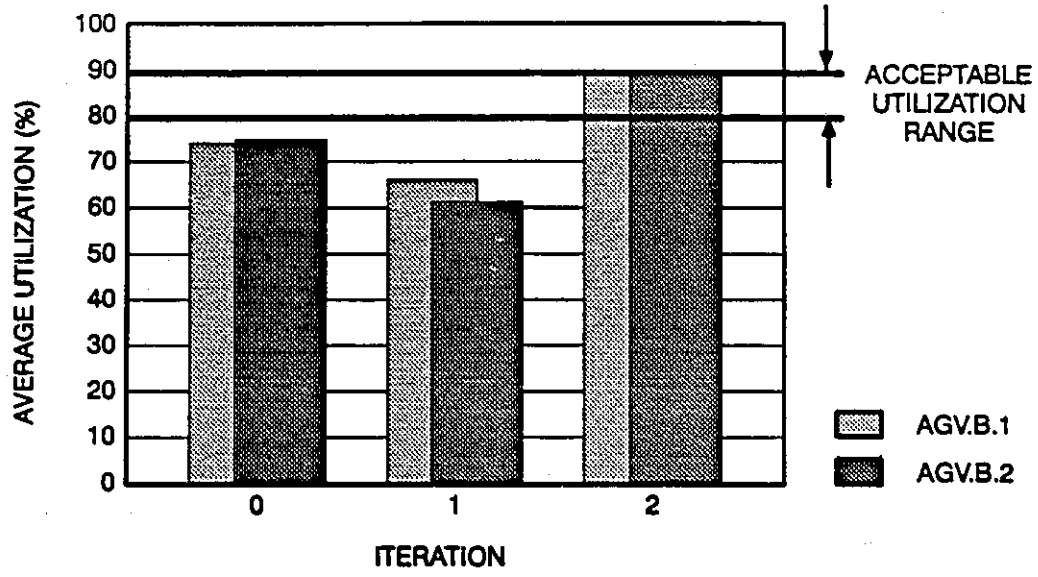


Figure 7.30. Average utilization of AGVs

simulation model was changed to reflect the changes suggested by the analyzer and was simulated.

7.2.3 Iteration 2

In iteration 2, there is a significant increase in the production volume of BLOCK 1 and BLOCK 2 (Figure 7.17). The production volume of BLOCK.1 was 338 and that of BLOCK.2 was 151 which is above the acceptable level of 333 and 147 units, respectively. There is a significant increase in the average utilization of all workstations. The average utilization of LOAD.A.1 and UNLOAD.A.1 is slightly less than 70% while that of NCMM.A.1, NCMM.A.2, and NCMM.A.3 is between 70% and 80% (Figure 7.18). This is due to an increase in the number of pallets circulating in the system.

There is a significant increase in the utilization of AGV.B.1 and AGV.B.2.

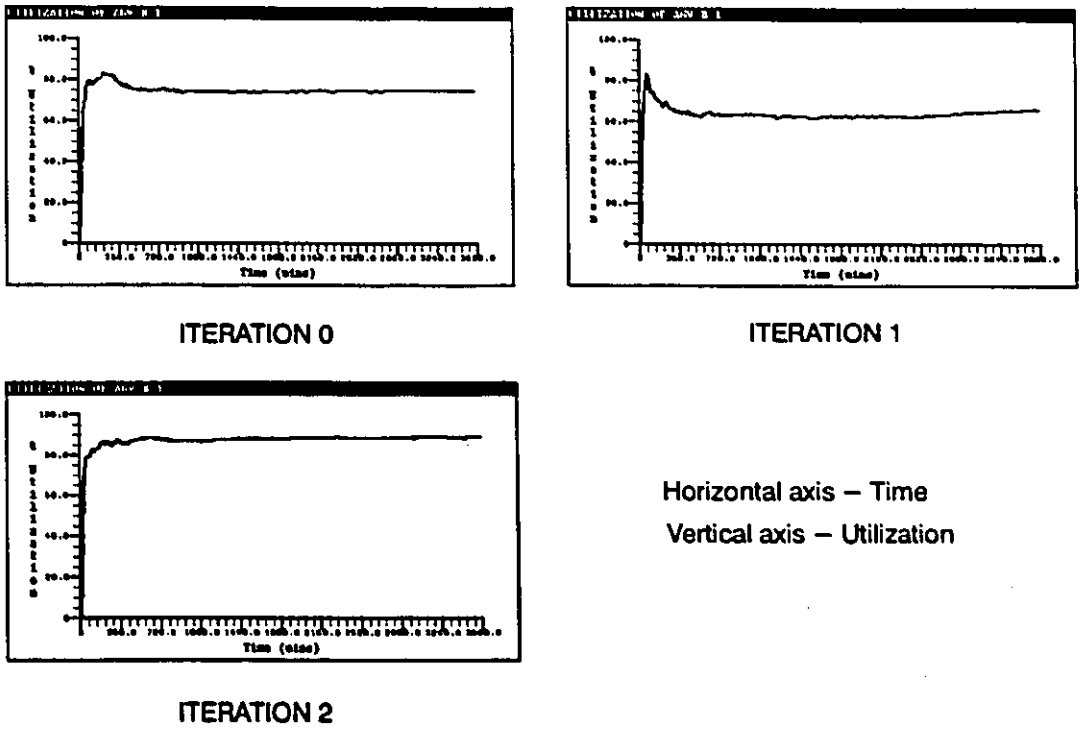


Figure 7.31. Utilization versus time of AGV.B.1

This again can be attributed to the increased number of pallets circulating in the system. The current utilizations of AGV.B.1 and AGV.B.2 are in the acceptable range of 80% to 90%. There is a small increase in the blocking percentage of the workstations (Figure 7.30) but this is still insignificant and well below the acceptable level of 10%.

The average queue length of pallets waiting for the workstation and the AGVs for each automatic pallet changer are shown in Figure 7.35, Figure 7.36, Figure 7.37, Figure 7.38, and Figure 7.39. From these plots it can be seen that the average queue lengths are well below the acceptable queue length of 75% for each automatic pallet changer.

At the end of iteration 2, all the performances measures are satisfied and

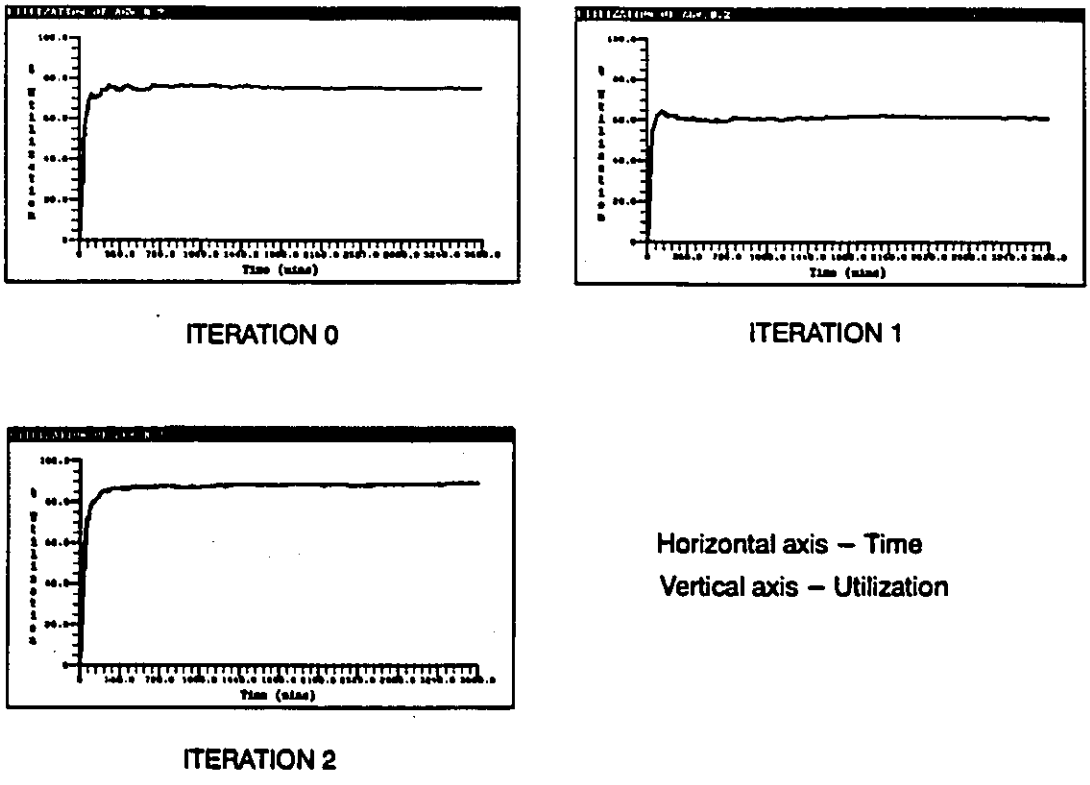


Figure 7.32. Utilization versus time of AGV.B.2

therefore, this design is the recommended final design. The investment cost index which is the ratio of the total investment to the parts produced within the payback period was calculated for each scenario. A payback period of five years was assumed for this case study. The investment cost index for all scenarios is shown in Figure 7.40. It can be found that this value decreases as the design is fine tuned which suggests that the design is getting better from initial to the final stage. The components of the final design are shown in Table 7.17. The graphical simulation model of the final design is the same as the starting one shown earlier in Figure 7.5.

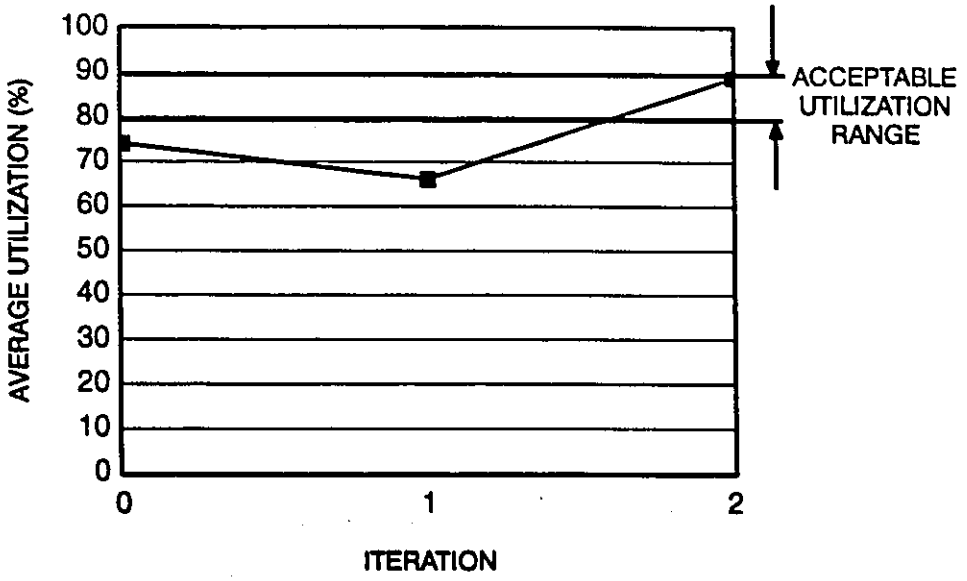


Figure 7.33. Average utilization of AGV.B.1

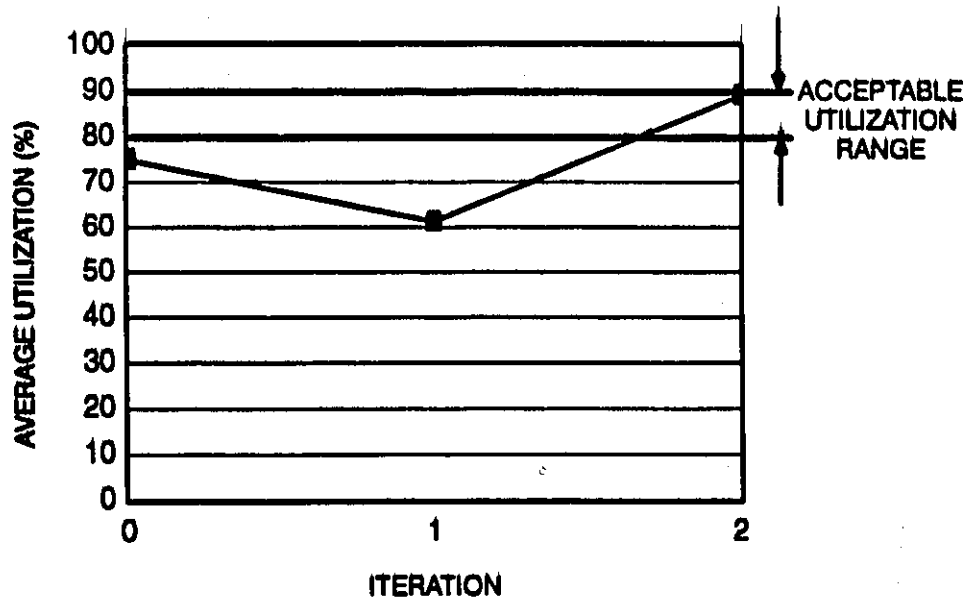


Figure 7.34. Average utilization of AGV.B.2

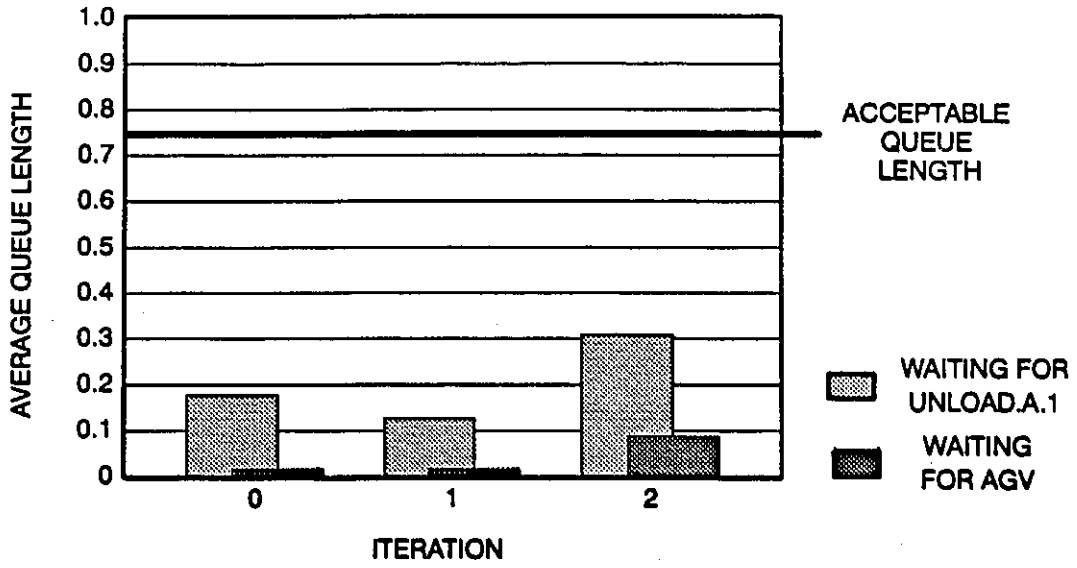


Figure 7.35. Average queue length of APC.2.B.1

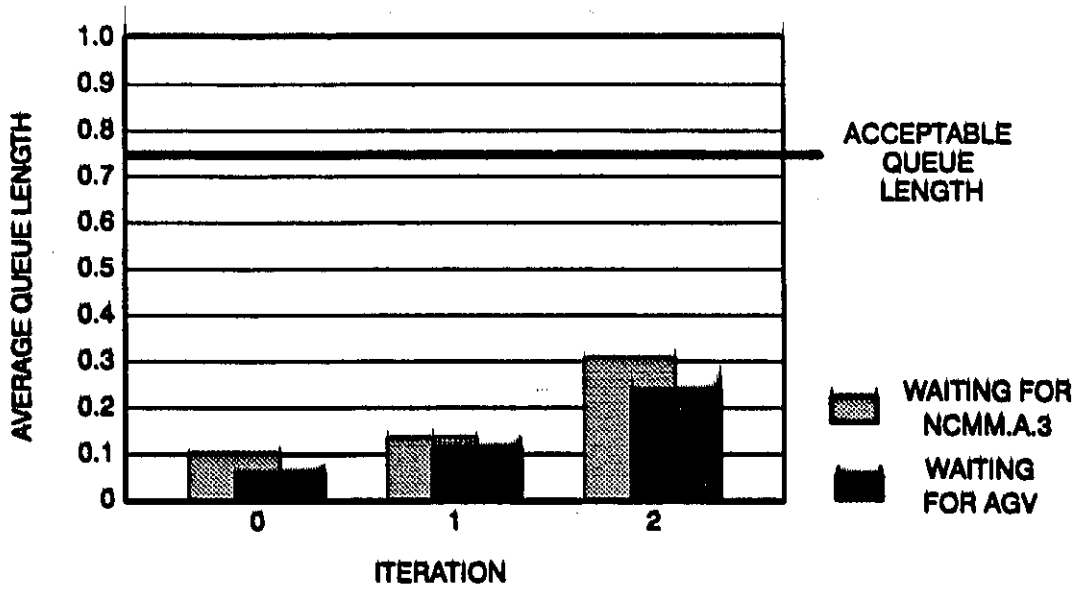


Figure 7.36. Average queue length of APC.2.B.2

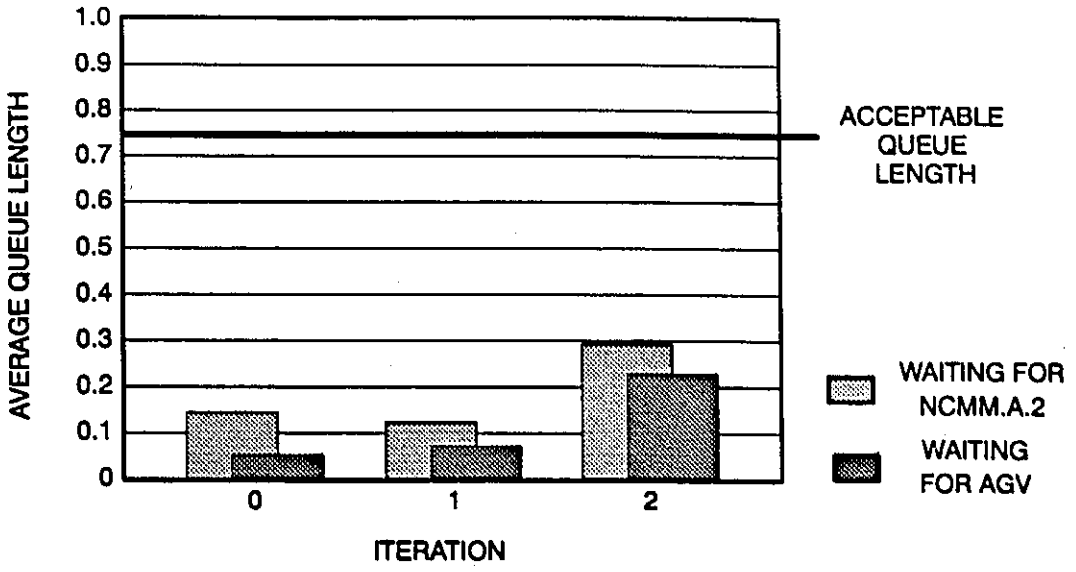


Figure 7.37. Average queue length of APC.2.B.3

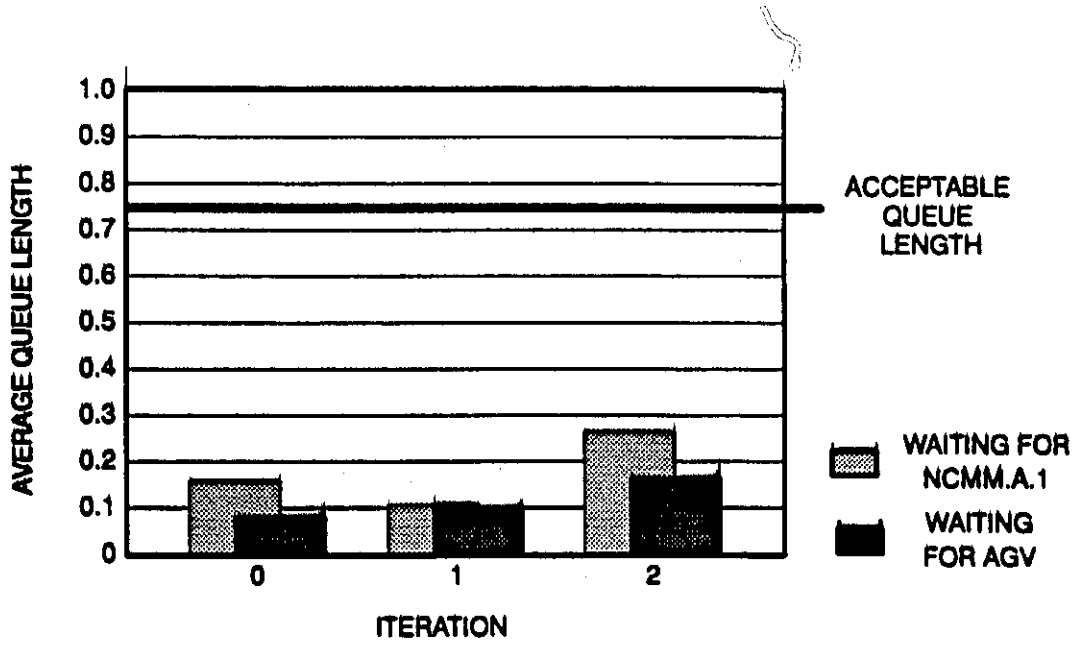


Figure 7.38. Average queue length of APC.2.B.4

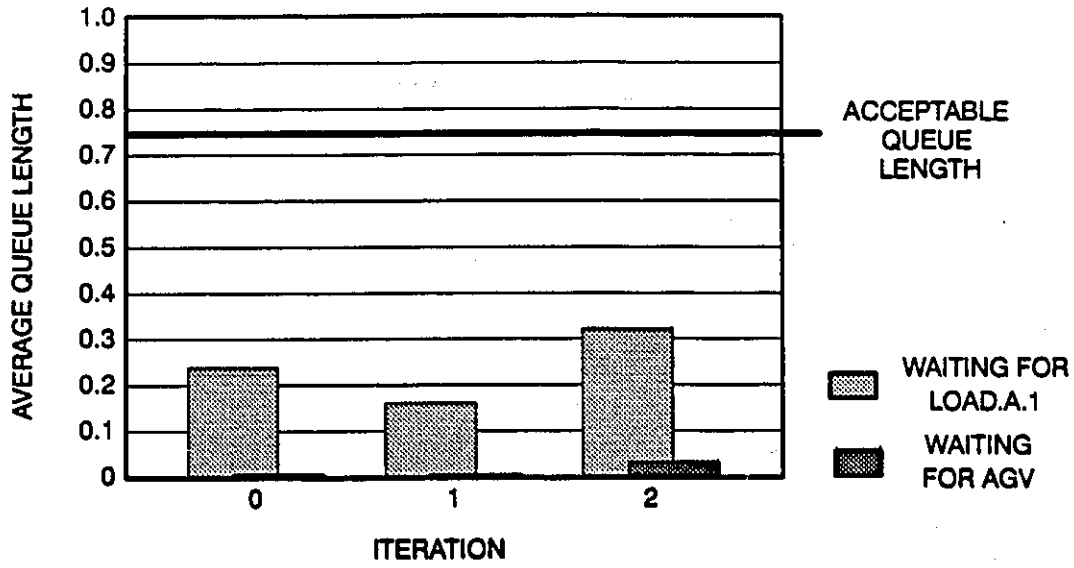


Figure 7.39. Average queue length of APC.2.B.5

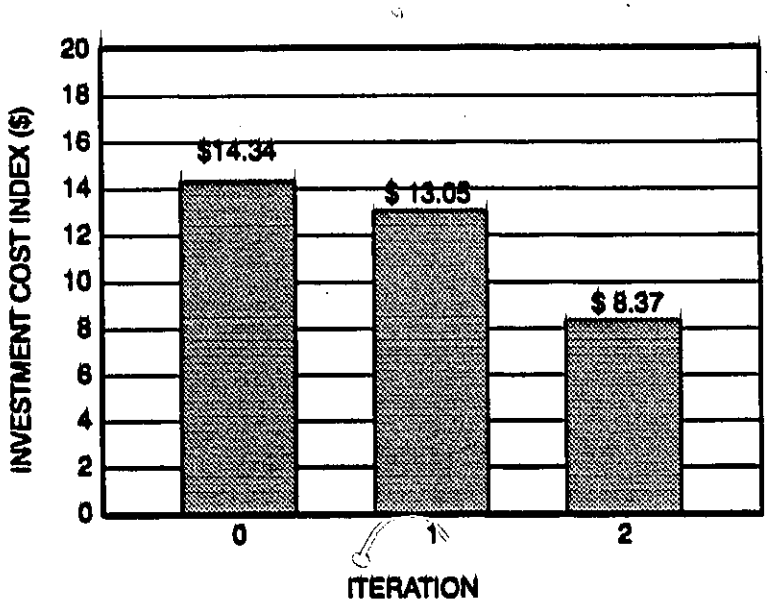


Figure 7.40. Investment cost per part produced over five years

Table 7.17. Components of final design for case study 1

Components	Component names
Pallets	PALLET.B.1, PALLET.B.2, PALLET.B.3, PALLET.B.4, PALLET.B.5, PALLET.B.6, PALLET.C.1, PALLET.C.2
Workstations	LOAD.A.1, NCMM.A.1, NCMM.A.2, NCMM.A.3, UNLOAD.A.1
Material handling systems	AGV.B.1, AGV.B.2
Automatic pallet changers	APC.2.B.1, APC.4.B.2, APC.4.B.3, APC.2.B.4, APC.2.B.5

7.3 Case study 2 – Fine tuning the original design

The performance of the FMS design developed in case study 2 (Figure 7.14) was then evaluated by simulating the model. The data collected during the simulation was also analyzed by the analyzer. This iterative procedure of developing the simulation model, simulating the model, and analyzing the results was repeated automatically by FMX. Three iterations were required for this case study. In the first iteration, the number of PALLET.C which carries both brake drums was increased by 19. In the second iteration, the number of PALLET.C was further increased by 10. In the third iteration, the speed of all the conveyor segments that make up the conveyor loop were increased to 3.25 meters/minute. These changes are listed in Table 7.18.

In flexible manufacturing systems where conveyors are used for transporting pallets between workstations, the conveyor is not a constraint in the system unless there is a sufficient number of pallets to fill up all the conveyor segments. In a situation where the number of pallets circulating in the system is very small, increasing the conveyor speed would not significantly increase the throughput. On the contrary, increasing the number of pallets in the system increases its throughput. Hence, in this case study the number of pallets of both

pallet types were increased before increasing the speed of the conveyor. The conflict resolution rule which increases the speed of an AGV or rail cart before increasing the number of pallets or AGVs does not fire when conveyors are used as the material handling system.

Table 7.18. Design changes during each iteration for case study 2

Iteration	Changes made
1	Increase PALLET.C by 19 (PALLET.C.6 through PALLET.C.24)
2	Increase PALLET.C by 10 (PALLET.C.25 through PALLET.C.34)
3	Increase speed of all conveyor segments to 3.25 meters/minute

The production volumes of BRAKE DRUM 1 and BRAKE DRUM 2 in a time of 600 minutes (a 10 hour shift) are 300 and 200, respectively. However, a minimum production volume of 95% was set to be achieved for both part types. This acceptable production volume amounts to 285 for BRAKE DRUM 1 and 190 for BRAKE DRUM 2.

The production volumes of both brake drums for each iteration are shown in Figure 7.41. The expected utilization of each workstation is between 65% and 75%. The average utilization of each workstation for each iteration is shown collectively in Figure 7.42. None of the workstations were blocked for any of the three iterations.

7.3.1 Iteration 0

Iteration 0 corresponds to the initial design generated by the design synthesizer. After the simulation it was found that both brake drums did not meet their acceptable production volumes (Figure 7.41) and all the workstations were

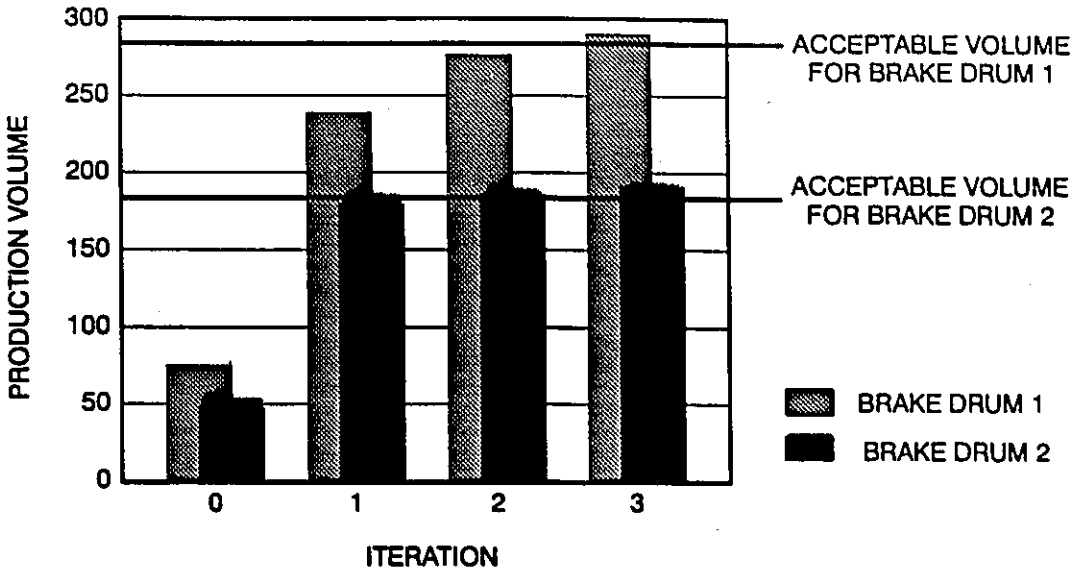


Figure 7.41. Production volumes of part types

under–utilized (Figure 7.42). None of the workstations were blocked and pallets did not wait long for a workstation or conveyor. The analyzer, therefore, increased the quantity of of PALLET.C, which carries BRAKE DRUM 1 and BRAKE DRUM 2, by 19.

7.3.2 Iteration 1

In iteration 1, there is a significant increase in the production volumes of BRAKE DRUM 1 and BRAKE DRUM 2 (Figure 7.41). This is mainly due to the increase in the number of pallets carrying these brake drums. The production volume of BRAKE DRUM 1 increased from 75 to 239 while that of BRAKE DRUM 2 increased from 48 to 180. These volumes are still below the acceptable production volumes of 285 and 190 units. Although the workstation utilizations increased, they are still below the lower limit of 65%. Again no workstation was

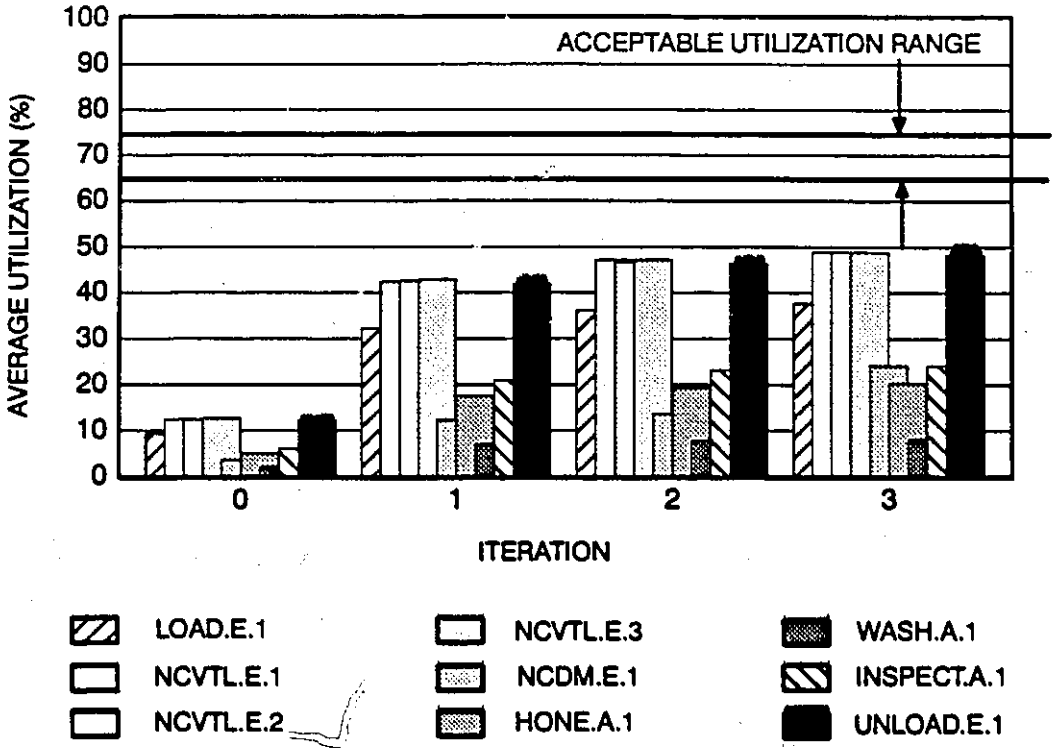


Figure 7.42. Average utilization of workstations

blocked. The analyzer, after this analysis, increased the quantity of PALLET.C by 10.

7.3.3 Iteration 2

In this iteration, there is an increase in the production volume of BRAKE DRUM 1 (from 239 to 277 units) and a very small increase in that of BRAKE DRUM 2 (from 180 to 186 units) as shown in Figure 7.41. Although, these volumes are very close to the acceptable production volume, they are still a little less. There is a very small increase in the workstation utilizations and again all the workstations are under–utilized (Figure 7.42). Also, none of the workstations were blocked. The

analyzer increased to speed of all the conveyor segments, CONVEYOR.1 through CONVEYOR.11, to 3.25 meters/minute from 3.00 meters/minutes.

7.3.4 Iteration 3

In this iteration, the production volumes for BRAKE DRUM 1 and BRAKE DRUM 2 are 291 and 191 units (Figure 7.41), respectively. This is greater than the acceptable production volumes. None of the workstations were blocked and hence, the average queue length of pallets waiting for a workstation or conveyor is less than the preset limit of 75%. As can be seen from Figure 7.42 all the workstations are under–utilized (less than 65% utilized). The analyzer, however, did not decrease their quantity because the processes in the process sequence of both brake drums need these workstations.

At the end of iteration 3, although the utilization of each workstation is less than the desired value, the analyzer stopped because the production volumes of both brake drums exceeded their acceptable volumes. Therefore, this design is the recommended final design. The investment cost index which is the ratio of the total investment to the parts produced within the payback period was calculated for each scenario. A payback period of three years was assumed for this case study and the investment cost index for all scenarios is shown in Figure 7.43. Like the previous case study it can be found that this value decreases as the design is fine tuned which suggests that the design is getting better from the initial to the final stage. The components of the final design for this case study are shown in Table 7.20.

7.3.5 Performance comparison

A simulation model of the actual manufacturing system designed by the company engineers was developed using the SIMAN simulation language. Values

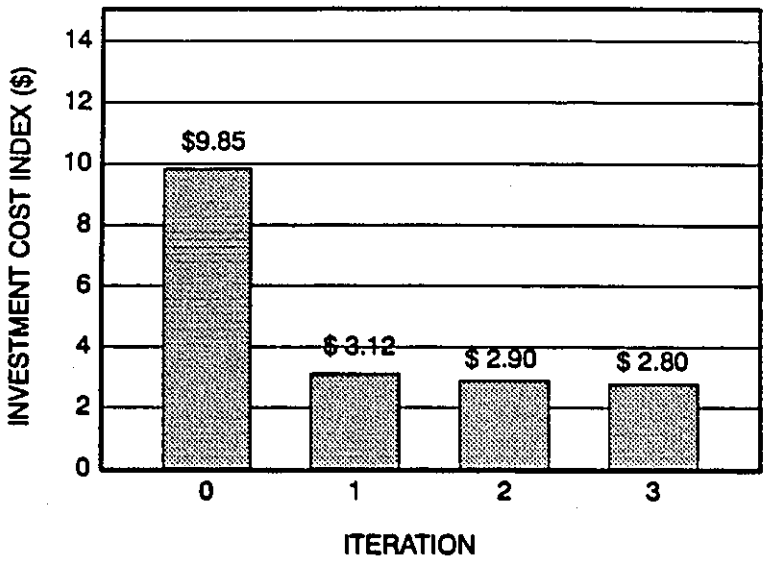


Figure 7.43. Investment cost per part produced over three years

of the various performance measures obtained from the SIMAN model were then compared with those from the third iteration of FMX. These are listed in Table 7.19. The total part production in 10 hours by the design synthesized and fine tuned by FMX indicates an increase from 440 units to 482 units—an increase of 9%. Also, the utilizations of the various workstations in the system generated by FMX are significantly higher than those from the SIMAN model of the actual system. This clearly demonstrates the effective decision support capability of FMX.

7.4 Discussion

The two case studies presented in the previous sections highlight the effectiveness of FMX. These case studies demonstrate the capability of the design synthesizer module of FMX to respond to variations in input data. The case study in Section 7.1.1.4 and Section 7.1.2.1 clearly shows the ability of FMX to handle

larger problems with more than two part types. The two major case studies illustrate the potential of FMX as a comprehensive decision support tool that reduces the FMS design cycle time and also increases the productivity of system designers.

Table 7.19. Performance comparison for case study 2

Performance measure	SIMAN output	FMX output
Total part production in 10 hours	440	482
LOAD.E.1 utilization	22.4%	37.7%
NCVTL.E.1 utilization	40.0%	48.9%
NCVTL.E.2 utilization	37.6%	49.0%
NCVTL.E.3 utilization	39.2%	49.0%
NCDM.E.1 utilization	20.0%	24.2%
HONE.A.1 utilization	20.0%	20.4%
WASH.A.1 utilization	4.0%	8.1%
INSPECT.A.1 utilization	18.4%	24.2%
UNLOAD.E.1 utilization	32%	48.2%

Table 7.20. Components of final design for case study 2

Components	Component names
Pallets	PALLET.C.1 through PALLET.C.34
Workstations	LOAD.E.1, NCVTL.E.1, NCVTL.E.2, NCVTL.E.3, NCDM.E.1, HONE.A.1, WASH.A.1, INSPECT.A.1, UNLOAD.E.1
Material handling systems	CONVEYOR.C.1, CONVEYOR.C.2, CONVEYOR.C.3, CONVEYOR.C.4, CONVEYOR.C.5, CONVEYOR.C.6, CONVEYOR.C.7, CONVEYOR.C.8, CONVEYOR.C.9, CONVEYOR.C.10, CONVEYOR.C.11
Automatic pallet changers	APC.2.C.1, APC.2.C.2, APC.2.C.3, APC.2.C.4, APC.2.C.5, APC.2.C.6, APC.2.C.7, APC.2.C.8, APC.2.C.9

CHAPTER 8

CONCLUSIONS

In this chapter, the conclusions of this research effort are presented. The various features of FMX which make it a comprehensive decision support tool are described in detail. Then the achievements of this work are summarized and FMX is compared with other similar systems. Finally, the limitations of this work are highlighted and the areas which should be addressed in the future to overcome these limitations are explained.

8.1 FMX features

Flexible manufacturing systems are a relatively new breed of automated and integrated manufacturing systems, primarily for the mid–volume and mid–variety type of batch manufacturing environment. The complex nature and the expensive investment incurred by these systems has made the design process very crucial for their successful performance after installation. FMS design is a complex and iterative process consisting of the following major activities: planning, design synthesis, model development, and output analysis.

Several approaches such as analytical, physical, and simulation models are available for modeling and evaluating the performance of FMS designs. Of these, simulation models have gained a wide acceptance because they:

(i) are more flexible and adaptable than physical models, and

(ii) capture more realistic information than analytical models.

Simulation is a computer–intensive process and there are several FMS simulators available for simulating a wide variety of systems. Simulation, however, encompasses other activities which make it a time–consuming and a difficult process. According to Shannon [43]:

"In order to use simulation correctly and intelligently, the designer is required to have expertise in number of different fields. This generally means separate courses in probability, statistics, design of experiments, modeling, computer programming, and a simulation language. This translates into 720 hours of formal classroom instructions plus another 1440 hours of outside study (more than 1 man–year of effort) and that is only to gain the basic tools."

When simulation is used for modeling and evaluating FMS designs, there are three major activities that are still done by humans. These include: design synthesis, simulation model development, and output analysis. System designers use their knowledge and experience for these front– and back–end activities of simulation. The manual nature of these difficult activities increases the FMS design cycle time.

The objective of this research was to automate the FMS design process by combining expert systems and discrete–event simulation. A prototype knowledge–based system called FMX (Flexible Manufacturing Expert) was developed to achieve this objective. This system combines object–oriented programming, expert system techniques, and simulation to automate the FMS design process. FMX was implemented using KEE and SimKit and has been applied to the machining domain. The various modules of FMX include the component

library, design synthesizer, simulation model developer, simulator, analyzer, and user interface.

8.1.1 Component library

The component library is object–oriented and contains a hierarchical description of both domain independent and domain dependent objects. This library was organized into three smaller libraries, SIMKIT, QLIB, and FMSLIB, to make it more manageable. The SIMKIT library contains objects necessary for discrete–event simulation such as the simulation clock, the simulation calendar, random number generators, and data collectors; the QLIB library contains objects for simulating queueing networks such as entities, servers, queues, and resources; and the FMSLIB library contains objects of a machining FMS such as workstations, material handling systems, pallets, and buffers. To develop a simulation model, the workstations were modeled as servers, buffers as queues, material handling systems and pallets as resources, and part types as entities.

The object–oriented approach was used in the implementation because of the following advantages:

(i) the simulated objects represent real–world objects such as FMS components,

(ii) the objects' properties and behaviors could be easily represented by attribute slots and method slots, and

(iii) the inheritance and encapsulation features of object–oriented programs make them highly reusable and easy to maintain.

8.1.2 Design synthesizer

The flexible manufacturing system design synthesis process requires two types of information:

(i) information about the family of part types to be processed, and

(ii) information about the system requirements.

The design synthesizer module of FMX is an expert system which contains knowledge used by system designers for generating initial FMS designs. It contains heuristic knowledge in the form of rules and analytical knowledge in the form of methods. These rules were organized into various rule classes and rules units so that each rule class solved a sub−problem such as selecting the workstations or selecting the material handling systems. This reduces the search space because only those rules that are needed to solve a certain sub−problem are fired. A forward chaining or a data−oriented strategy was used to fire the rules in the synthesizer. These rules were written using the TellAndAsk syntax of KEE and the methods were written in LISP.

The design generated by the design synthesizer is not the final design, but is a starting point which leads towards the final FMS design. This is done iteratively by evaluating the performance of design every time there is a change. The design which satisfies all its performance measures is the final design that is recommended for installation. It was found in both case studies discussed in Chapter 7 that the initial design generated by the design synthesizer was always under capacity and components had to be added in subsequent steps to improve its performance. This is because at this point in the design process, the generated design is solely based on static information without considering the dynamics of the system.

The output from the design synthesizer is the list of selected components

that make up the FMS, the connections between these components, and the approximate positions of these components in the layout. This information is then fed to the simulation model developer which generates a graphical simulation model.

8.1.3 Model developer and simulator

The output from the design synthesizer is a list of components that comprise the flexible manufacturing system. A simulation model of this design is then developed to evaluate its performance using discrete–event simulation techniques. This model is developed on a computer using one of the several commercially available simulation languages. Irrespective of the simulation language, model development is a time–consuming process and requires the system designer to be experienced and knowledgeable in the chosen simulation. Moreover, an increase in complexity or level of detail in a model directly increases the model development time.

The FMX prototype developed in this thesis has a model developer module for automatically generating simulation models. This module performs the following functions:

(i) It establishes the various relationships between the system components.

(ii) It develops a graphical display of the model using the icons associated with the components in the component library.

(iii) It attaches data collectors to the various components in the design for collecting appropriate data such as average utilization of workstations, average queue length of pallets waiting for a workstation or material handling system, and blocking of workstations.

(iv) It attaches plots to data collectors for plotting values of variables over time

such as plotting the utilization of a workstation with respect to time. This module performs these functions by using methods which are attached to slots of objects. These methods are activated by sending messages to the appropriate slot.

The simulator module of FMX executes the simulation model of an FMS design by stepping through the various events that take place in the system over time. It involves the following steps: initializing the model before each simulation run, animating the model, and calculating average values at the end of the simulation run. FMX has sophisticated animation in that it displays the movement of parts in the system and the state of the different stationary system components. Several functions were written in LISP to animate the simulation.

8.1.4 Analyzer

Data generated after every simulation run has to be analyzed to evaluate the performance of the design. Traditionally this has been the responsibility of the system designers who peruse enormous data, both numeric and graphic, to identify the deficiencies in a given design. Using their knowledge and experience, they then recommend appropriate design changes to overcome these deficiencies. The manual nature of the output analysis process makes it difficult and time-consuming.

The analyzer module of FMX developed in this thesis is a diagnostic expert system which analyzes the data obtained from the simulation to evaluate the performance of the current FMS design. It uses the following performance measures as guidelines to evaluate designs:

(i) part production rates,

(ii) average equipment utilizations,

(iii) average equipment blocking, and

(iv) average queue lengths of automatic pallet changers.

The analyzer performs the following major functions: compares the values of performance measures obtained from the simulation model with the preset values, diagnoses the problems, identifies design deficiencies, and recommends design changes to overcome these deficiencies. In situations when more than one recommendation is made, the analyzer uses conflict resolution strategies to choose one recommendation that will have a significant impact on the performance of the system and is cost effective.

The analyzer contains the knowledge of system designers in the form of rules or methods. Like the design synthesizer, these rules were organized into various rule classes and rule units. A backward chaining or a goal–oriented strategy was used to fire the rules in the analyzer. These rules were written using the TellAndAsk syntax of KEE and the methods were written in LISP. The output from the analyzer is a recommended design change. This information is input to the simulation model developer which generates a model of the new design for the next iteration.

8.2 Summary of achievements

The simulation phase of the FMS design process has always been the focus of many researchers ([22], [24], [25], [26], [27], [28], and [29]). However, there are activities prior to and after simulation that are still done manually by experienced system designers. The use of expert systems for these knowledge–intensive activities is still in its early stages and the research efforts until now have been isolated and rudimentary. Researchers have overlooked the common features of expert systems and discrete–event simulation in developing a comprehensive decision support system that combines both techniques effectively.

The knowledge–based system, FMX, developed in this thesis combines object–oriented programming, expert system techniques, and discrete–event simulation methodology to design new flexible manufacturing systems. FMX is a comprehensive system which uses: (i) expert system techniques to generate designs, develop simulation models, and analyze output; (ii) discrete–event simulation methodology to evaluate the performance of FMS designs; and (iii) an object–oriented programming paradigm.

The most significant achievement in this thesis was the development of this comprehensive decision support system for designing FMS. To the author's knowledge there is no single system which integrates all phases of the FMS design process in a single system. Each module of FMX addresses a particular phase of the design process and is a contribution by itself. These contributions can be summarized as follows:

(i) the development of an expert design synthesizer to generate initial flexible manufacturing systems design which satisfies stated functional requirements,

(ii) the development of a simulation model developer to convert FMS designs to graphical simulation models automatically,

(iii) the development of an expert analyzer for automatically analyzing the simulation output, identifying deficiencies in the current design, and recommend changes to the current system design, and

(iv) the integration of the design synthesizer, simulation model developer, simulator, and output analyzer in a single software framework for an effective and comprehensive decision support tool.

8.3 Future extensions

FMX is a comprehensive knowledge–based system for designing and evaluating flexible manufacturing systems. It consists of modules each addressing a phase of the design process such as the design synthesizer for design synthesis and the analyzer for output analysis. The case studies illustrated in Chapter 7 prove the potential of FMX for reducing the FMS design cycle time and generating better and consistent designs. Since it is impossible to consider every detail in one thesis, some areas which have to be addressed in the future to enhance the performance of FMX are as follows:

(i) Flexible manufacturing can be used in a wide variety of manufacturing processes such as assembly, machining, molding, sheet metal forming, and welding. Currently, FMX has been applied for designing flexible manufacturing systems in the machining environment. The methodology or principles used in this application can be extended to the other manufacturing domains as well. This would mean an increase in the number and type of components (e.g., workstations, tools, etc.) in the component library and the number of rules and methods in the other modules to include those machines or expert rules specific to the new domains.

(ii) Any expert system is never complete and can be improved constantly by adding knowledge to the knowledge base which enhances the performance of the system and provides better results. The design synthesizer and analyzer modules of FMX are basically expert systems which contain heuristic and analytical knowledge for expert problem solving. The knowledge bases of these modules can be expanded by adding more rules and methods to enhance their performance. For example, the design

synthesizer selects workstations based on important features such as dimensions, process capability, accuracy, and special tool requirements. This selection process could be further refined by considering other features such as the capacity of the tool magazine of a workstation. More rules would be required for this new line of reasoning and updating this knowledge in the current implementation of FMX requires proficiency in LISP, KEE, and SimKit. Similarly, the analyzer could include additional rules to analyze the simulation output.

(iii) Only two types of layouts, line and loop, were considered in this work. But there are other types of layouts which are more complex than these two. Future extension to FMX should consider other types of layouts such as the tree or network layouts.

(iv) FMX considers only design changes to improve the performance of a flexible manufacturing system. However, significant improvements in FMS performance can be achieved sometimes by changing operational procedures such as scheduling rules. The current implementation of FMX considers only the FIFO (First-In-First-Out) scheduling rule for palletized parts waiting for workstations or material handling system. This is adequate for the initial system design. Future extension to FMX could incorporate other scheduling rules which enables FMX to be used for the day to day operation of the system, not just the initial design.

(v) FMX is a deterministic system and does not have any variability built into it. Hence, a single simulation run is enough to get an accurate value of the various performance measures. This is sufficient for initial systems design. Future extensions to FMX could have the capability to model stochastic systems (for example, randomly distributed processing times).

(vi) Currently, FMX does not model equipment breakdowns. In reality, however, machines always breakdown and remain in that state for some time. Also, preventive maintenance of workstations or material handling systems increases the equipment down time. To handle these issues, the future extension of FMX should consider this aspect. However, it is felt that such issues are more important for the day to day operation of FMS, not their initial design.

REFERENCES

[1] Saul, G.,1985, "Flexible Manufacturing System is CIM Implemented at the Shop Floor Level," *Industrial Engineering*, Vol. 17, No. 6, pp. 35–39.

[2] Hartley, J., 1984, *FMS at Work*, IFS (Publications) Ltd., U.K.

[3] Market Intelligence Research Corporation, 1991, "Bright Outlook for FMS and Cells," *Manufacturing Engineering*, July, pp. 18–20.

[4] ElMaraghy, H.A., 1982, "Simulation and Graphical Animation of Advanced Manufacturing Systems," *Journal of Manufacturing Systems*, Vol. 1, No. 1, pp. 53–63.

[5] Rembold, U., Blume, C., and Dillmann, R., *Computer–Integrated Manufacturing Technology and Systems*, Marcel Dekker, Inc., New York, U.S.A., pp. 742–776.

[6] Maher, M.L., 1990, "Process Models for Design Synthesis," *AI Magazine*, Winter, pp. 49–58.

[7] Parunak, H.V.D., Kindrick, J.D., and Muralidhar, K., 1989, "MAPCon: An Expert System with Multiple Reasoning Objectives," *Knowledge–Based Systems in Manufacturing*, A. Kusiak, ed., Taylor & Francis Ltd., Great Britain.

[8] Seliger, G., Viehweger, B., and Kommana, S.R, 1988, "Integrated Planning of Manufacturing Systems," *Robotics and Computer Integrated Manufacturing,* No. 3/4, pp. 593–600.

[9] Parthasarathy, S.T., and Kim, S.H., 1991, "Design of Intelligent Manufacturing Systems: Critical Decision Structures and Performance Metrics," *Artificial Intelligence in Design,* D. T. Pham, ed., Springer Verlag, Great Britain, pp. 465–491.

[10] ElMaraghy, H.A., and Ravi, T., 1992, "Modern Tools for the Design, Modelling, and Evaluation of Flexible Manufacturing Systems", *Robotics & Computer Integrated Manufacturing,* Vol. 9, N0. 4/5, pp. 335–340.

[11] Buzacott, J.A., and Shantikumar, J.G., 1980, "Models for Understanding Flexible Manufacturing Systems," *AIIE Transactions*, Vol. 12, No. 4, pp. 339–350.

[12] Suri, R., 1983, "Robustness of Queueing Network Formulae," *Journal of the ACM,* Vol. 30, No. 5, pp. 564–594.

[13] Tay, Y.C., 1985, "Error Bounds for Performance Prediction in Queueing Networks," *ACM Transactions on Computer Systems,* Vol. 3, No. 3, pp. 227–254.

[14] Suri, R., Diehl, G., and Dean, R., 1986, "Quick and Easy Manufacturing Analysis Using MANUPLAN," *FMS Current Issues and Models,* F. Choobineh and R. Suri, ed., Institute of Industrial Engineers, U.S.A.

[15] Brown, E., 1988, "IBM Continues Rapid Modeling Technique and Simulation to Design PCB Factory of the Future," *Industrial Engineering,* Vol. 20, No. 6, pp. 23–26.

[16] Huettner, C.M., and Steudel, H.S., 1992, "Analysis of a Manufacturing System via Spreadsheet Analysis, Rapid Modeling, and Manufacturing Simulation," *International Journal of Production Research*, Vol. 30, No. 7, pp. 1699–1714.

[17] Gupta, T., 1993, "Design of Manufacturing Cells for Flexible Environment Considering Alternative Routing," *International Journal of Production Research*, Vol. 31, No. 6, pp. 1259–1273.

[18] Nof, S.Y., Deisenroth, M.P., and Meier, W.L., 1980, "Computerized Physical Simulators are Developed to Solve IE Problems," *Industrial Engineering*, Vol. 12, No. 10, pp. 70–75.

[19] Diesch, K.H., and Malstrom, E.M., 1985, "Physical Simulator Analyzes Performance of Flexible Manufacturing System," *Industrial Engineering*, Vol. 17, No. 6, pp. 66–77.

[20] Choi, R.H., and Malstrom, E.M., 1988, "Evaluation of Traditional Work Scheduling Rules in a Flexible Manufacturing System with a Physical Simulator," *Journal of Manufacturing Systems*, Vol. 7, No. 1, pp. 33–46.

[21] Turner, D.H., 1986, "Manufacturing Simulation Comes of Age," *CIM Technology*, Fall, pp. 16–19.

[22] Law, A.M., and Haider, S.W., 1989, "Selecting Simulation Software for Manufacturing Applications – Practical Guidelines and Software Survey," *Industrial Engineering*, Vol. 21, No. 5, pp. 33–46.

[23] O'Grady, P.J., and Menon, U., 1986, "A Concise Review of Flexible Manufacturing Systems and FMS Literature," *Computers in Industry*, Vol. 7, No. 2, pp. 155–167.

[24] Lenz, J.E., and Talavage, J.J., 1977, "General Computerized Manufacturing System Simulator," *Report No. 7, NSF Grant No. 74 15256,* School of Industrial Engineering, Purdue University, West Lafayette.

[25] ElMaraghy, H.A., and Ho, N.C., 1982, "A Flexible System for Computer Control of Manufacturing," *Computers in Mechanical Engineering (CIME)*, Vol. 1, No. 1, pp. 17–23.

[26] Stute, G., Storr, A., and Chmielnicki, S., 1982, "Planning of Flexible Systems—Simulation and Motion Display on a Graphic CRT," *Manufacturing Systems,* Vol. 12, No. 2, pp. 130–136.

[27] Crite, G. D., Mills, R.J., and Talavage, J.J., 1985, "PATHSIM – A Modular Simulator for Automatic Tool Handling System Evaluation in FMS," *Journal of Manufacturing Systems*, Vol. 4, No. 1, pp. 15–28.

[28] Eversheim, W., 1987, "Graphical Interactive Simulation for the Planning of Manufacturing Systems," *Journal of Manufacturing Systems*, Vol. 6, No. 2, pp. 151–156.

[29] Montazeri, M., Gelders, L.F., and Van Wassenhove, L.N., 1988, "A Modular Simulator for Design, Planning, and Control of Flexible Manufacturing Systems," *The International Journal of Advanced Manufacturing Technology*, Vol. 3, No. 1, pp. 15–32.

[30] Lenz, J.E., and Wichmann, K.E., 1986, "Expert Systems for FMS Design," *Technical Paper MS86–173,* Society of Manufacturing Engineers, Dearborn, U.S.A.

[31] Wadhwa, S., Felix, C., and Browne, J., 1987, "A Goal Directed Data Driven Simulator for FAS Design," *Proceedings of the European Simulation Conference*, July 8 – 10.

[32] Haddock, J., 1987, "An Expert System Framework Based on a Simulation Generator," *Simulation*, Vol. 48, No. 2, pp. 45–53.

[33] Mellichamp, J. M., and Wahab, A. F. A., 1987, "An Expert System for FMS Design," *Simulation*, Vol. 48, No. 5, pp. 201–208.

[34] Floss, P., and Talavage, J., 1990, "A Knowledge–Based Design Assistant for Intelligent Manufacturing Systems," *Journal of Manufacturing Systems*, Vol. 9, No. 2, pp. 87–102.

[35] Solberg, J.J., 1978, "Quantitative Design Tools for Computerized Manufacturing Systems," *Proceedings of the North American Metalworking Research Conference*, pp. 409–413.

[36] Mellichamp, J. M., Kwon, O., and Wahab, A. F. A., 1990, "FMS Designer: An Expert System for Flexible Manufacturing System Design," *International Journal of Production Research*, Vol. 28, No. 11, pp. 2013–2024.

[37] Forgy, C., 1985, *The OPS83 User's Manual*, Production System Technologies, Pittsburg, Pennsylvania.

[38] Frenzel, L. E., 1987, *Crash Course in Artificial Intelligence and Expert Systems*, Howard. W. Sams & Co., A Division of Macmillan, Inc., U. S. A., pp. 1–24.

[39] Nebendahl, D., 1988, *Expert Systems – Introduction to the Technology and Applications*, John Wiley & Sons Ltd., Great Britain.

[40] OKK Machines advertisement, *Manufacturing Engineering*.

[41] IntelliCorp, 1988, *KEETutor – Module 8 and Module 9*, IntelliCorp, Inc., U.S.A.

[42] O'Keefe, R., 1986, "Simulation and Expert Systems – A Taxonomy and Some Examples," *Simulation*, Vol. 46, No. 1, pp. 10–16.

[43] Shannon, R. E., 1988, "Knowledge Based Simulation Techniques for Manufacturing," *International Journal of Production Research*, Vol. 26, No. 5, pp. 953–973.

[44] IntelliCorp, 1988, *KEE User's Guide*, IntelliCorp, Inc., U.S.A.

[45] IntelliCorp, 1986, *The SimKit System User Manual*, IntelliCorp, Inc., U.S.A.

[46] Stecke, K. E., and Browne, J., 1985, "Variations in Flexible Manufacturing Systems According to the Relevant Types of Automated Materials Handling," *Material Flow*, Vol. 2, Nos. 2 & 3, pp. 179–185.

[47] Gabbert, P., and Brown, D.E., 1987, "A Knowledge–based Approach to Materials Handling System Design in Manufacturing Facilities," *Proceedings of the World Productivity Forum and International Industrial Engineering Conference*, Washington, D.C., pp. 445.

[48] Leskowsky, Z., Logan, L., and Vanelli, A., 1987, "Group Technology Decision Aids in an Expert System for Plant Layout," *Modern Production Management Systems*, A. Kusiak, ed., North–Holland, Amsterdam, pp. 561.

[49] Abdou, G., and Dutta, S.P., 1990, "An Integrated Approach to Facilities Layout Using Expert Systems," *International Journal of Production Research*, Vol. 28, No. 4, pp. 685–708.

[50] Ohmi, T., 1984, "Material Flow and its Flexibility in Flexible Manufacturing Systems," *Proceedings of the 1st International Machine Tool Engineers Conference,* Tokyo, Japan, pp. 166–177.

[51] IntelliCorp, 1988, *TellAndAsk Reference Manual, IntelliCorp*, Inc., U.S.A.

[52] Byrd, T.A., 1992, "Implementation and Use of Expert System in Organizations: Perceptions of Knowledge Engineers," *Journal of Management Information Systems,* Vol. 8, No. 4, pp. 97–116.

[53] Hoffman, R., 1987, "The Problem of Extracting the Knowledge of Experts from the Perspective of Experimental Psychology," *AI Magazine,* Vol. 8, No. 2, pp. 53–67.

[54] Hayes–Roth, F., Waterman, D., and Lenat, D., 1983, *Building Expert Systems,* Addison–Wesley Publishing Company, Inc., U.S.A.

[55] Boose, J., 1986, "ETS: A System for the Transfer of Human Expertise," *Knowledge Based Problem Solving,* J. Kowalik, ed., Englewood Cliffs, New Jersey, U.S.A.

[56] Abdul–Gader, A.H., and Kozar, K.A., 1990, "Discourse Analysis for Knowledge Acquisition: The Coherence Method," *Journal of Management Information Systems,* Vol. 6, No. 4, pp. 61–82.

[57] Agarwal, R., and Tanniru, M., 1990, "Knowledge Acquisition Using Structured Interviewing: An Empirical Investigation," *Journal of Management Information Systems,* Vol. 7, No. 1, pp. 123–140.

[58] Nance, R.E., 1984, "Model Development Revisited," *Proceedings of the 1984 Winter Simulation Conference,* Dallas, Texas, pp. 75–80.

[59] Murray, K.J., and Sheppard, S.V., 1988, "Knowledge–based Simulation Model Specification," *Simulation,* Vol 50, No. 3, pp. 112–119.

[60] Pegden, C., Shannon, R.E., and Sadowski, R.P., 1990, *Introduction to Simulation using SIMAN*, McGraw–Hill, Inc., U.S.A.

[61] Ho, N.C., 1981, "Discrete Simulation of Flexible Manufacturing System," *Master's Thesis*, McMaster University, Hamilton, Canada.

APPENDIX A

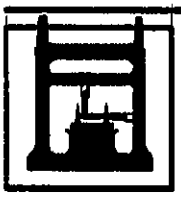
THE COMPONENT LIBRARY – FMSLIB

One of the modules of Flexible Manufacturing Expert (FMX) is the component library which contains a hierarchical description of both domain–independent and domain dependent objects. FMSLIB is a sublibrary of the SIMKIT and QLIB super libraries. It is an object–oriented library which contains the various domain–dependent objects of flexible manufacturing systems in the machining domain such as workstations, material handling systems, automatic pallet changers, and pallets. In this appendix, the attributes of each class of objects, their values, and the icons associated with these objects are shown. The size of the icon is proportional to actual foot print dimensions of the component. The information about each object shown on the following pages is an actual display presented to the user by FMX when the user requests information about an object in FMSLIB.

A.1 Workstations

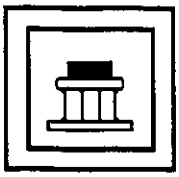
Name of workstation: HONE.A
Description: Large-sized CNC honing machine

Length of workstation: 375 cms
Breadth of workstation: 375 cms
Size of workstation table: 50X50 cms
Maximum height accommodated: 30 cms
Clearance of workstation: 150 cms
Cost of workstation: $120000
Maximum load capacity: 400 kgms
Accuracy of workstation: HIGH
Special tools supported: NO
Processes: (HONE)
Process times: (0.25) minutes



Name of workstation: INSPECT.A
Description: Large-sized CMM inspection station

Length of workstation: 375 cms
Breadth of workstation: 375 cms
Size of workstation table: 50X50 cms
Maximum height accommodated: 30 cms
Clearance of workstation: 150 cms
Cost of workstation: $75000
Maximum load capacity: 400 kgms
Accuracy of workstation: HIGH
Special tools supported: NO
Processes: (INSPECT-1)
Process times: (0.3) minutes



Name of workstation: LOAD.A

Description: Large-sized loading station

Length of workstation: 375 cms
Breadth of workstation: 375 cms
Size of workstation table: 50X50 cms
Maximum height accommodated: 30 cms
Clearance of workstation: 150 cms
Cost of workstation: $50000
Maximum load capacity: 400 kgms
Accuracy of workstation: LOW
Special tools supported: NO
Processes: (LOAD-1 LOAD-2)
Process times: (6 8.35) minutes

Name of workstation: LOAD.B

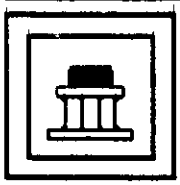
Description: Medium-sized loading station

Length of workstation: 250 cms
Breadth of workstation: 250 cms
Size of workstation table: 40X40 cms
Maximum height accommodated: 20 cms
Clearance of workstation: 100 cms
Cost of workstation: $40000
Maximum load capacity: 300 kgms
Accuracy of workstation: LOW
Special tools supported: NO
Processes: (LOAD-1 LOAD-2)
Process times: (5.5 8.4) minutes

Name of workstation: LOAD.C

Description: Small-sized loading station

Length of workstation: 125 cms
Breadth of workstation: 125 cms
Size of workstation table: 30X30 cms
Maximum height accommodated: 10 cms
Clearance of workstation: 50 cms
Cost of workstation: $30000
Maximum load capacity: 200 kgms
Accuracy of workstation: LOW
Special tools supported: NO
Processes: (LOAD-1 LOAD-2)
Process times: (5 0.5) minutes



Name of workstation: LOAD.B

Description: Large-sized loading station

Length of workstation: 375 cms
Breadth of workstation: 375 cms
Size of workstation table: 50X50 cms
Maximum height accommodated: 30 cms
Clearance of workstation: 150 cms
Cost of workstation: $75000
Maximum load capacity: 400 kgms
Accuracy of workstation: HIGH
Special tools supported: NO
Processes: (LOAD-1 LOAD-2)
Process times: (5 0.37) minutes

Name of workstation: LOAD.E

Description: Medium-sized loading station

Length of workstation: 250 cms
Breadth of workstation: 250 cms
Size of workstation table: 40X40 cms
Maximum height accommodated: 20 cms
Clearance of workstation: 100 cms
Cost of workstation: $65000
Maximum load capacity: 300 kgms
Accuracy of workstation: HIGH
Special tools supported: NO
Processes: (LOAD-1 LOAD-2)
Process times: (4.5 0.44) minutes

Name of workstation: LOAD.F

Description: Small-sized loading station

Length of workstation: 125 cms
Breadth of workstation: 125 cms
Size of workstation table: 30X30 cms
Maximum height accommodated: 10 cms
Clearance of workstation: 50 cms
Cost of workstation: $55000
Maximum load capacity: 200 kgms
Accuracy of workstation: HIGH
Special tools supported: NO
Processes: (LOAD-1 LOAD-2)
Process times: (4 0.52) minutes

Name of workstation: MCDM.A

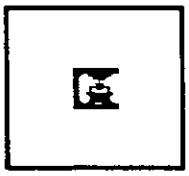
Description: Large-sized CNC drilling machine

Length of workstation: 375 cms
Breadth of workstation: 375 cms
Size of workstation table: 50X50 cms
Maximum height accommodated: 30 cms
Clearance of workstation: 150 cms
Cost of workstation: $300000
Maximum load capacity: 400 kgms
Accuracy of workstation: LOW
Special tools supported: YES
Processes: (DRILL-1 DRILL-2 DRILL-3 REAM-1 REAM-2 REAM-3)
Process times: (7.5 2 0.2 6 1.5 0.2) minutes

Name of workstation: MCDM.B

Description: Medium-sized CNC drilling machine

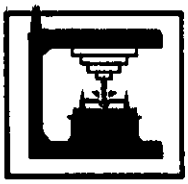
Length of workstation: 250 cms
Breadth of workstation: 250 cms
Size of workstation table: 40X40 cms
Maximum height accommodated: 20 cms
Clearance of workstation: 100 cms
Cost of workstation: $200000
Maximum load capacity: 300 kgms
Accuracy of workstation: LOW
Special tools supported: YES
Processes: (DRILL-1 DRILL-2 DRILL-3 REAM-1 REAM-2 REAM-3)
Process times: (7 2 0.17 5.5 1.5 0.17) minutes



Name of workstation: NCDM.C

Description: Small-sized CNC drilling machine

Length of workstation: 125 cms
Breadth of workstation: 125 cms
Size of workstation table: 30X30 cms
Maximum height accommodated: 10 cms
Clearance of workstation: 50 cms
Cost of workstation: $100000
Maximum load capacity: 200 kgms
Accuracy of workstation: LOW
Special tools supported: YES
Processes: (DRILL-1 DRILL-2 DRILL-3 REAM-1 REAM-2 REAM-3)
Process times: (6.5 1.5 0.15 5 1.5 0.15) minutes



Name of workstation: NCDM.B

Description: Large-sized CNC drilling machine

Length of workstation: 375 cms
Breadth of workstation: 375 cms
Size of workstation table: 50X50 cms
Maximum height accommodated: 30 cms
Clearance of workstation: 150 cms
Cost of workstation: $350000
Maximum load capacity: 400 kgms
Accuracy of workstation: HIGH
Special tools supported: YES
Processes: (DRILL-1 DRILL-2 DRILL-3 DRILL-4 REAM-1 REAM-2 REAM-3)
Process times: (6.2 4.5 0.21 0.23 5.5 4 0.21) minutes

Name of workstation: NCBM.E

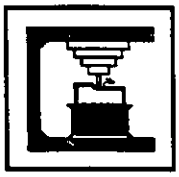
Description: Medium-sized CNC drilling machine

Length of workstation: 250 cms
Breadth of workstation: 250 cms
Size of workstation table: 40X40 cms
Maximum height accommodated: 20 cms
Clearance of workstation: 100 cms
Cost of workstation: $225000
Maximum load capacity: 300 kgms
Accuracy of workstation: HIGH
Special tools supported: YES
Processes: (DRILL-1 DRILL-2 DRILL-3 DRILL-4 REAM-1 REAM-2 REAM-3)
Process times: (6 4.5 0.17 0.18 5 3.5 0.17) minutes

Name of workstation: NCBM.F

Description: Small-sized CNC drilling machine

Length of workstation: 125 cms
Breadth of workstation: 125 cms
Size of workstation table: 30X30 cms
Maximum height accommodated: 10 cms
Clearance of workstation: 50 cms
Cost of workstation: $125000
Maximum load capacity: 200 kgms
Accuracy of workstation: HIGH
Special tools supported: YES
Processes: (DRILL-1 DRILL-2 DRILL-3 DRILL-4 REAM-1 REAM-2 REAM-3)
Process times: (5.5 4 0.15 0.17 5 3.5 0.15) minutes



Name of workstation: MCNC.A

Description: Large-sized CNC milling machine

Length of workstation: 375 cms
Breadth of workstation: 375 cms
Size of workstation table: 50X50 cms
Maximum height accommodated: 30 cms
Clearance of workstation: 150 cms
Cost of workstation: $400000
Maximum load capacity: 400 kgms
Accuracy of workstation: LOW
Special tools supported: NO
Processes: (MILL-1 MILL-2 DRILL-1 DRILL-2 REAM-1 REAM-2)
Process times: (9 7 9 5.5 7 5.5) minutes

Name of workstation: MCNC.B

Description: Medium-sized CNC milling machine

Length of workstation: 250 cms
Breadth of workstation: 250 cms
Size of workstation table: 40X40 cms
Maximum height accommodated: 20 cms
Clearance of workstation: 100 cms
Cost of workstation: $325000
Maximum load capacity: 300 kgms
Accuracy of workstation: LOW
Special tools supported: NO
Processes: (MILL-1 MILL-2 DRILL-1 DRILL-2 REAM-1 REAM-2)
Process times: (7.5 6.5 8.5 5.5 6.5 5) minutes

Name of workstation: MCNM.C

Description: Small-sized CNC milling machine

Length of workstation: 125 cms
Breadth of workstation: 125 cms
Size of workstation table: 30X30 cms
Maximum height accommodated: 10 cms
Clearance of workstation: 50 cms
Cost of workstation: $250000
Maximum load capacity: 200 kgms
Accuracy of workstation: LOW
Special tools supported: NO
Processes: (MILL-1 MILL-2 DRILL-1 DRILL-2 REAM-1 REAM-2)
Process times: (0 6 0 5 6 5) minutes

Name of workstation: MCNM.B

Description: Large-sized CNC milling machine

Length of workstation: 375 cms
Breadth of workstation: 375 cms
Size of workstation table: 50X50 cms
Maximum height accommodated: 30 cms
Clearance of workstation: 150 cms
Cost of workstation: $450000
Maximum load capacity: 400 kgms
Accuracy of workstation: HIGH
Special tools supported: YES
Processes: (MILL-1 MILL-2 DRILL-1 DRILL-2 REAM-1 REAM-2)
Process times: (0 6.5 0 2.5 6.5 2) minutes

Name of workstation: MCNX.E

Description: Medium-sized CNC milling machine

Length of workstation: 250 cms
Breadth of workstation: 250 cms
Size of workstation table: 40X40 cms
Maximum height accommodated: 20 cms
Clearance of workstation: 100 cms
Cost of workstation: $350000
Maximum load capacity: 300 kgms
Accuracy of workstation: HIGH
Special tools supported: YES
Processes: (MILL-1 MILL-2 DRILL-1 DRILL-2 REAM-1 REAM-2)
Process times: (7.5 6 7.5 2 6 2) minutes

Name of workstation: MCNX.F

Description: Small-sized CNC milling machine

Length of workstation: 125 cms
Breadth of workstation: 125 cms
Size of workstation table: 30X30 cms
Maximum height accommodated: 10 cms
Clearance of workstation: 50 cms
Cost of workstation: $275000
Maximum load capacity: 200 kgms
Accuracy of workstation: HIGH
Special tools supported: YES
Processes: (MILL-1 MILL-2 DRILL-1 DRILL-2 REAM-1 REAM-2)
Process times: (7 6 7 1.5 6 1.5) minutes

Name of workstation: NCVTL.A

Description: Large-sized CNC vertical turret lathe

Length of workstation: 375 cms
Breadth of workstation: 375 cms
Size of workstation table: 50X50 cms
Maximum height accommodated: 30 cms
Clearance of workstation: 150 cms
Cost of workstation: $350000
Maximum load capacity: 400 kgms
Accuracy of workstation: LOW
Special tools supported: NO
Processes: (TURN-1 BORE-1 BORE-2 BORE-3)
Process times: (0.62 6.5 0.52 0.63) minutes

Name of workstation: NCVTL.B

Description: Medium-sized CNC vertical turret lathe

Length of workstation: 250 cms
Breadth of workstation: 250 cms
Size of workstation table: 40X40 cms
Maximum height accommodated: 20 cms
Clearance of workstation: 100 cms
Cost of workstation: $275000
Maximum load capacity: 300 kgms
Accuracy of workstation: LOW
Special tools supported: NO
Processes: (TURN-1 BORE-1 BORE-2 BORE-3)
Process times: (0.6 6 0.54 0.65) minutes

Name of workstation: NCVTL.C

Description: Small-sized CNC vertical turret lathe

Length of workstation: 125 cms
Breadth of workstation: 125 cms
Size of workstation table: 30X30 cms
Maximum height accommodated: 10 cms
Clearance of workstation: 50 cms
Cost of workstation: $150000
Maximum load capacity: 200 kgms
Accuracy of workstation: LOW
Special tools supported: NO
Processes: (TURN-1 BORE-1 BORE-2 BORE-3)
Process times: (0.68 5.5 0.58 0.7) minutes

Name of workstation: NCVTL.B

Description: Large-sized CNC vertical turret lathe

Length of workstation: 375 cms
Breadth of workstation: 375 cms
Size of workstation table: 50X50 cms
Maximum height accommodated: 30 cms
Clearance of workstation: 150 cms
Cost of workstation: $400000
Maximum load capacity: 400 kgms
Accuracy of workstation: HIGH
Special tools supported: NO
Processes: (TURN-1 BORE-1 BORE-2 BORE-3)
Process times: (0.6 5 0.5 0.61) minutes

Name of workstation: MCVTL.E

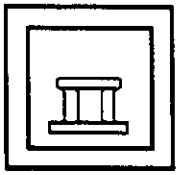
Description: Medium-sized CNC vertical turret lathe

Length of workstation: 250 cms
Breadth of workstation: 250 cms
Size of workstation table: 40X40 cms
Maximum height accommodated: 20 cms
Clearance of workstation: 100 cms
Cost of workstation: $325000
Maximum load capacity: 300 kgms
Accuracy of workstation: HIGH
Special tools supported: NO
Processes: (TURN-1 BORE-1 BORE-2 BORE-3)
Process times: (0.62 7.2 0.53 0.64) minutes

Name of workstation: MCVTL.F

Description: Small-sized CNC vertical turret lathe

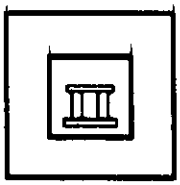
Length of workstation: 125 cms
Breadth of workstation: 125 cms
Size of workstation table: 30X30 cms
Maximum height accommodated: 10 cms
Clearance of workstation: 50 cms
Cost of workstation: $175000
Maximum load capacity: 200 kgms
Accuracy of workstation: HIGH
Special tools supported: NO
Processes: (TURN-1 BORE-1 BORE-2 BORE-3)
Process times: (0.66 6 0.57 0.67) minutes



Name of workstation: UNLOAD.A

Description: Large-sized unloading station

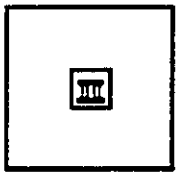
Length of workstation: 375 cms
Breadth of workstation: 375 cms
Size of workstation table: 50X50 cms
Maximum height accommodated: 30 cms
Clearance of workstation: 150 cms
Cost of workstation: $50000
Maximum load capacity: 400 kgms
Accuracy of workstation: LOW
Special tools supported: NO
Processes: (UNLOAD-1 UNLOAD-2)
Process times: (6 0.5) minutes



Name of workstation: UNLOAD.B

Description: Medium-sized unloading station

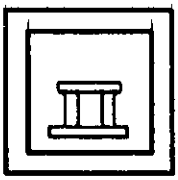
Length of workstation: 250 cms
Breadth of workstation: 250 cms
Size of workstation table: 40X40 cms
Maximum height accommodated: 20 cms
Clearance of workstation: 100 cms
Cost of workstation: $40000
Maximum load capacity: 300 kgms
Accuracy of workstation: LOW
Special tools supported: NO
Processes: (UNLOAD-1 UNLOAD-2)
Process times: (5.5 0.6) minutes



Name of workstation: UNLOAD.C

Description: Small-sized unloading station

Length of workstation: 125 cms
Breadth of workstation: 125 cms
Size of workstation table: 30X30 cms
Maximum height accommodated: 10 cms
Clearance of workstation: 50 cms
Cost of workstation: $30000
Maximum load capacity: 200 kgms
Accuracy of workstation: LOW
Special tools supported: NO
Processes: (UNLOAD-1 UNLOAD-2)
Process times: (5 0.67) minutes



Name of workstation: UNLOAD.B

Description: Large-sized unloading station

Length of workstation: 375 cms
Breadth of workstation: 375 cms
Size of workstation table: 50X50 cms
Maximum height accommodated: 30 cms
Clearance of workstation: 150 cms
Cost of workstation: $75000
Maximum load capacity: 400 kgms
Accuracy of workstation: HIGH
Special tools supported: NO
Processes: (UNLOAD-1 UNLOAD-2)
Process times: (5 0.5) minutes

Name of workstation: UNLOAD.E

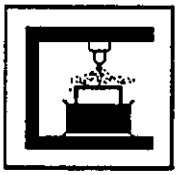
Description: Medium-sized unloading station

Length of workstation: 250 cms
Breadth of workstation: 250 cms
Size of workstation table: 40X40 cms
Maximum height accommodated: 20 cms
Clearance of workstation: 100 cms
Cost of workstation: $65000
Maximum load capacity: 300 kgms
Accuracy of workstation: HIGH
Special tools supported: NO
Processes: (UNLOAD-1 UNLOAD-2)
Process times: (4.5 0.6) minutes

Name of workstation: UNLOAD.F

Description: Small-sized unloading station

Length of workstation: 125 cms
Breadth of workstation: 125 cms
Size of workstation table: 30X30 cms
Maximum height accommodated: 10 cms
Clearance of workstation: 50 cms
Cost of workstation: $55000
Maximum load capacity: 200 kgms
Accuracy of workstation: HIGH
Special tools supported: NO
Processes: (UNLOAD-1 UNLOAD-2)
Process times: (4 0.67) minutes

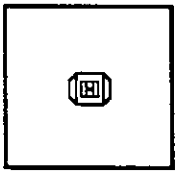


Name of workstation: WASH.A

Description: Large-sized washing station

Length of workstation: 375 cms
Breadth of workstation: 375 cms
Size of workstation table: 50X50 cms
Maximum height accommodated: 30 cms
Clearance of workstation: 150 cms
Cost of workstation: $20000
Maximum load capacity: 400 kgms
Accuracy of workstation: HIGH
Special tools supported: NO
Processes: (WASH)
Process times: (0.1) minutes

A.2 Material handling systems



Name of material handling system: AGV.A

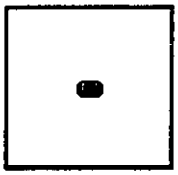
Description: Large-sized automated guided vehicle

Length of MHS: 125 cms
Breadth of MHS: 100 cms
Maximum pallet size carried: 50X50 cms
Adjacent distance between rows: 300 cms
Clearance for MHS travel: 15 cms
Maximum weight carried: 400 kgms
Cost of MHS: $80000

Name of material handling system: AGV.B

Description: Medium-sized automated guided vehicle

Length of MHS: 100 cms
Breadth of MHS: 75 cms
Maximum pallet size carried: 40X40 cms
Adjacent distance between rows: 240 cms
Clearance for MHS travel: 10 cms
Maximum weight carried: 300 kgms
Cost of MHS: $60000



Name of material handling system: AGV.C

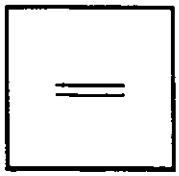
Description: Small-sized automated guided vehicle

Length of MHS: 75 cms
Breadth of MHS: 50 cms
Maximum pallet size carried: 30X30 cms
Adjacent distance between rows: 180 cms
Clearance for MHS travel: 5 cms
Maximum weight carried: 200 kgms
Cost of MHS: $40000

Name of material handling system: CONVEYOR.A

Description: Large-sized conveyor

Length of MHS: Not applicable
Breadth of MHS: 50 cms
Maximum pallet size carried: 50X50 cms
Adjacent distance between rows: 300 cms
Clearance for MHS travel: 15 cms
Maximum weight carried: Not applicable
Cost of MHS: $5 per cm



Name of material handling system: CONVEYOR.B

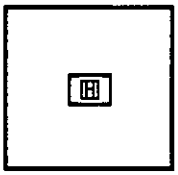
Description: Medium-sized conveyor

Length of MHS: Not applicable
Breadth of MHS: 40 cms
Maximum pallet size carried: 40X40 cms
Adjacent distance between rows: 240 cms
Clearance for MHS travel: 10 cms
Maximum weight carried: Not applicable
Cost of MHS: $4 per cm

Name of material handling system: CONVEYOR.C

Description: Small-sized conveyor

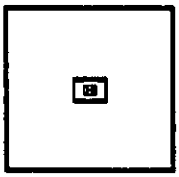
Length of MHS: Not applicable
Breadth of MHS: 30 cms
Maximum pallet size carried: 30X30 cms
Adjacent distance between rows: 180 cms
Clearance for MHS travel: 5 cms
Maximum weight carried: Not applicable
Cost of MHS: $3 per cm



Name of material handling system: RAIL.CART.A

Description: Large-sized rail cart

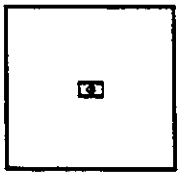
Length of MHS: 125 cms
Breadth of MHS: 100 cms
Maximum pallet size carried: 50X50 cms
Adjacent distance between rows: NIL cms
Clearance for MHS travel: 15 cms
Maximum weight carried: 400 kgms
Cost of MHS: $50000



Name of material handling system: RAIL.CART.B

Description: Medium-sized rail cart

Length of MHS: 100 cms
Breadth of MHS: 75 cms
Maximum pallet size carried: 40X40 cms
Adjacent distance between rows: NIL cms
Clearance for MHS travel: 10 cms
Maximum weight carried: 300 kgms
Cost of MHS: $40000

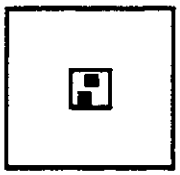


Name of material handling system: RAIL.CART.C

Description: Small-sized rail cart

Length of MHS: 75 cms
Breadth of MHS: 50 cms
Maximum pallet size carried: 30X30 cms
Adjacent distance between rows: NIL cms
Clearance for MHS travel: 5 cms
Maximum weight carried: 200 kgms
Cost of MHS: $30000

A.3 Buffers



Name of automatic pallet changer: APC.2.A

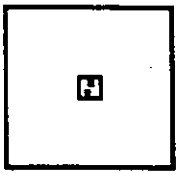
Description: Large-sized automatic pallet changer

Length of APC: 125 cms
Breadth of APC: 125 cms
Capacity of APC: 2 pallets
Maximum load: 800 kgms
Pallet size accommodated: 50X50 cms
Cost of APC: $40000

Name of automatic pallet changer: APC.2.B

Description: Medium-sized automatic pallet changer

Length of APC: 100 cms
Breadth of APC: 100 cms
Capacity of APC: 2 pallets
Maximum load: 600 kgms
Pallet size accommodated: 40X40 cms
Cost of APC: $30000



Name of automatic pallet changer: APC.2.C

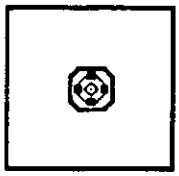
Description: Small-sized automatic pallet changer

Length of APC: 75 cms
Breadth of APC: 75 cms
Capacity of APC: 2 pallets
Maximum load: 400 kgms
Pallet size accommodated: 30X30 cms
Cost of APC: $20000

Name of automatic pallet changer: APC.4.A

Description: Large-sized automatic pallet changer

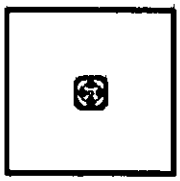
Length of APC: 165 cms
Breadth of APC: 165 cms
Capacity of APC: 4 pallets
Maximum load: 1600 kgms
Pallet size accommodated: 50X50 cms
Cost of APC: $50000



Name of automatic pallet changer: APC.4.B

Description: Medium-sized automatic pallet changer

Length of APC: 135 cms
Breadth of APC: 135 cms
Capacity of APC: 4 pallets
Maximum load: 1200 kgms
Pallet size accommodated: 48X48 cms
Cost of APC: $48000



Name of automatic pallet changer: APC.4.C

Description: Small-sized automatic pallet changer

Length of APC: 100 cms
Breadth of APC: 100 cms
Capacity of APC: 4 pallets
Maximum load: 800 kgms
Pallet size accommodated: 30X30 cms
Cost of APC: $38000

Name of automatic pallet changer: APC.6.A

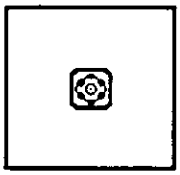
Description: Large-sized automatic pallet changer

Length of APC: 200 cms
Breadth of APC: 200 cms
Capacity of APC: 6 pallets
Maximum load: 2400 kgms
Pallet size accommodated: 50X50 cms
Cost of APC: $60000

Name of automatic pallet changer: APC.6.B

Description: Medium-sized automatic pallet changer

Length of APC: 165 cms
Breadth of APC: 165 cms
Capacity of APC: 6 pallets
Maximum load: 1800 kgms
Pallet size accommodated: 40X40 cms
Cost of APC: $50000

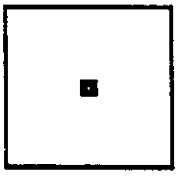


Name of automatic pallet changer: APC.6.C

Description: Small-sized automatic pallet changer

Length of APC: 125 cms
Breadth of APC: 125 cms
Capacity of APC: 6 pallets
Maximum load: 1200 kgms
Pallet size accommodated: 30X30 cms
Cost of APC: $40000

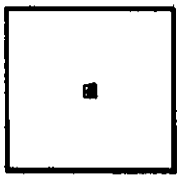
A.4 Pallets



Name of pallet: PALLET.A

Description: Large-sized pallet type

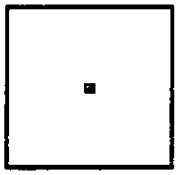
Dimensions of pallet: 50X50 cms
Height of pallet: 5 cms
Weight of pallet: 30 kgms
Cost of pallet: $10000



Name of pallet: PALLET.B

Description: Medium-sized pallet type

Dimensions of pallet: 40X40 cms
Height of pallet: 5 cms
Weight of pallet: 25 kgms
Cost of pallet: $7500



Name of pallet: PALLET.C

Description: Small-sized pallet type

Dimensions of pallet: 30X30 cms

Height of pallet: 5 cms

Weight of pallet: 10 kgms

Cost of pallet: $5000

APPENDIX B

MINIMZING TRAVEL DISTANCE

This appendix illustrates the positioning of workstations around a loop so that the travel distances are minimized. Seven workstations were used for this purpose and the various combinations of these seven workstations around the loop are shown in Figure B.1. Minimizing the travel distance in the loop requires these seven workstations to be split almost equally on either side of the loop. From the five layouts shown in Figure B.1, Layout 3 with three workstations on one side and four workstations on the other results in the minimum travel distance for this situation. An algorithm has been developed in FMX to generate workstation positions using this concept and is implemented as a method.

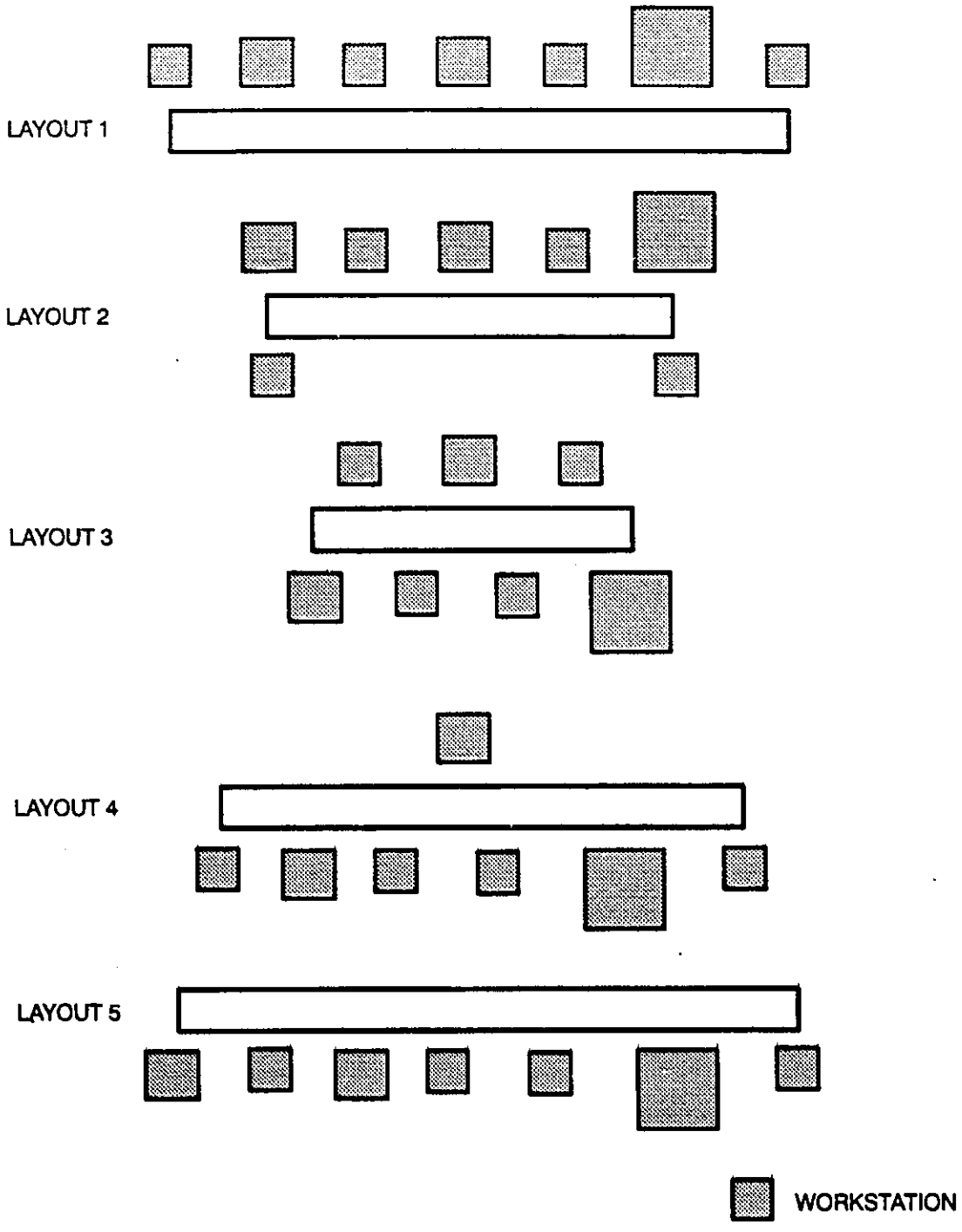
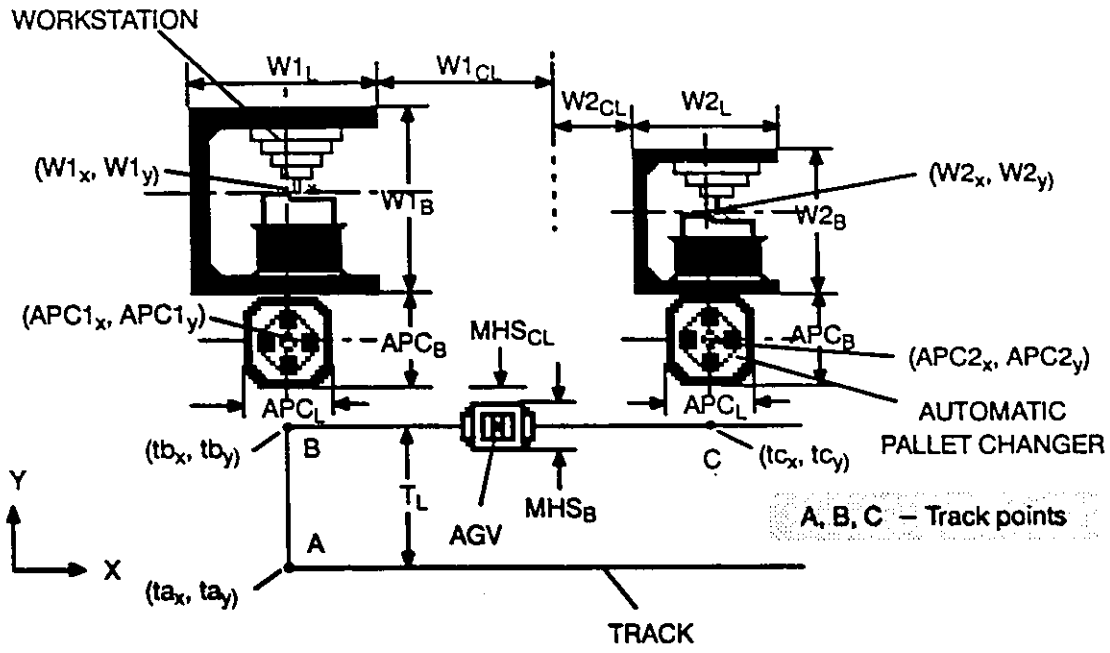


Figure B.1. Minimizing travel distance in a loop layout

APPENDIX C

COORDINATES OF SYSTEM COMPONENTS

The graphical simulation model of an FMS design is an actual representation of how the system looks from the top. The various system components such as workstations, material handling systems, and automatic pallet changers are positioned at specific coordinate positions with respect to one another. The simulation model developer uses information about the inter-relationships between the various system components, their dimensions, and clearances to generate a graphical simulation model. The relationships that are used to calculate the coordinate positions of the various system components are shown in Figure C.1.



$$W1_x = APC1_x = tb_x = ta_x$$
$$tb_y = ta_y + T_L$$
$$APC1_y = tb_y + MHS_{CL} + 1/2\ (MHS_B + APC_B)$$
$$W1_y = APC1_y + 1/2\ (APC_B + W1_B)$$
$$tc_x = tb_x + 1/2\ (W1_L + W2_L) + W1_{CL} + W2_{CL}$$
$$W2_x = APC2_x = tc_x$$
$$tc_y = tb_y$$
$$APC2_y = tc_y + MHS_{CL} + 1/2\ (MHS_B + APC_B)$$
$$W2_y = APC2_y + 1/2\ (APC_B + W2_B)$$

Figure C.1. Calculating the coordinates of system components