1

MICROD

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service

Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

TABLE OF CONTENTS (CONT'D)	<u>Page</u>
2.2.2 Incorporation.....	25
2.3 Summary.....	26
3.0 The Methodology With Parallelism.....	27
3.1 The Flow of the Design.....	28
3.2 Process Flow Issues.....	35
3.2.1 Stoppages.....	35
3.2.2 Resources.....	36
3.2.3 Changes in Requirements.....	38
3.2.4 False Technical Assumptions.....	39
3.2.5 Convergence.....	40
3.3 Summary.....	40
4.0 MANAGEMENT ISSUES AND OTHER ISSUES	
FACING A DESIGN.....	42
4.1 Economics of the Design.....	43
4.2 Allocation of Resources.....	45
4.3 Control Issues.....	47
4.4 The Role of the Customer.....	53
4.5 Quality Assurance.....	54

Permission has been granted to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film.

The author (copyright owner) has reserved other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without his/her written permission.

L'autorisation a été accordée à la Bibliothèque nationale du Canada de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur (titulaire du droit d'auteur) se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation écrite.

ISBN 0-315-54420-1

The undersigned hereby recommends to the Faculty of Graduate Studies and Research acceptance of the thesis,

**THE LOGICAL DESIGN OF AN
ENGINEERING PROJECT MANAGEMENT
SUPPORT SYSTEM**

submitted by William S. Samulack, B.Sc., in partial fulfilment of the requirements for the degree of Master of Science.

Thesis Supervisor

Chairman,
Department of Systems and
Computer Engineering

Department of Systems and Computer Engineering
Faculty of Engineering
Carleton University
May, 1987

ABSTRACT

System's design methodologies usually contain a sequence of activities separated by milestones. These milestones mark the completion of a set of activities which are assumed to have a strict precedence relationship to later design phases in a project. Milestones serve not only to bound these activities but, perhaps more importantly, provide a control mechanism for managing the project. In fact, based on a Manager's assessment of the risks involved, certain activities of a project often proceed without the completion of a previous milestone. This often permits a more effective utilization of resources resulting in lower costs. Such a practice appears to suffer from a loss of important control mechanisms. However, synchronization and control are achieved in practice by virtue of the Manager's experience.

This thesis addresses the two questions:

- Can the parallelism used in practice be formally modelled?
- and,
- Can such a model be controlled in a systematic way?

In response to these questions, a parallel version of a serial methodology is developed and a Kernel is designed which is shown to provide the necessary synchronization and control. This Kernel could be surrounded by a Management Expert System and offer a Project Manager a design aid that would allow him/her to have effective control of a project, to adjust the project to changing demands, and to have sufficient data to report its status.

ACKNOWLEDGEMENTS

I would like to thank Professor "Archie" Bowen for introducing me to the discipline of Structured Design. I would also like to thank him for his guidance and direction in the writing of this thesis and for his ability to ruthlessly spot errors or omissions throughout the text.

Thanks to Cindy for her perseverance and support throughout the years, especially during the writing of this thesis.

TABLE OF CONTENTS

	<u>Page</u>
Abstract.....	iii
Acknowledgments.....	iv
List of Illustrations.....	xi
1.0 Introduction.....	1
1.1 The System.....	7
1.2 The System Use.....	9
1.3 The Kernel.....	11
2.0 The Basic Methodology.....	14
2.1 Logical Design.....	14
2.1.1 Context Diagram.....	17
2.1.2 Data Flow Analysis.....	19
2.1.2.1 Data Flow Graph.....	20
2.1.2.2 Data Dictionary.....	21
2.1.3 Structure Diagrams.....	21
2.2 Physical Design.....	23
2.2.1 Preparation.....	23
2.2.1.1 Technology.....	23
2.2.1.2 Partitioning and Allocation.....	24

TABLE OF CONTENTS (CONT'D)

Page

2.2.2 Incorporation.....	25
2.3 Summary.....	26
3.0 The Methodology With Parallelism.....	27
3.1 The Flow of the Design.....	28
3.2 Process Flow Issues.....	35
3.2.1 Stoppages.....	35
3.2.2 Resources.....	36
3.2.3 Changes in Requirements.....	38
3.2.4 False Technical Assumptions.....	39
3.2.5 Convergence.....	40
3.3 Summary.....	40
4.0 MANAGEMENT ISSUES AND OTHER ISSUES	
FACING A DESIGN.....	42
4.1 Economics of the Design.....	43
4.2 Allocation of Resources.....	45
4.3 Control Issues.....	47
4.4 The Role of the Customer.....	53
4.5 Quality Assurance.....	54

TABLE OF CONTENTS (CONT'D)	<u>Page</u>
4.6 Production / Test.....	56
4.7 Documentation.....	60
4.7.1 Hardware Documentation.....	62
4.7.2 Software Documentation.....	62
4.7.3 Documentation Control.....	63
4.8 Logistics Support.....	63
4.9 Summary.....	64
5.0 THE PARALLEL MODEL VERSUS THE ISSUES.....	67
5.1 Process Flow Issues.....	68
5.2 Management Issues.....	69
5.3 Risk Factor.. .	71
5.4 Summary.....	74
6.0 THE SYSTEM.....	76
6.1 The User Interface.....	78
6.2 Program Management Tools.....	91
6.3 Summary.....	92
7.0 THE KERNEL.....	93
7.1 Requirements Definition.....	94
7.1.1 Design Process Requirements.....	94
7.1.2 Management Requirements.....	96
7.1.2 Other Requirements.....	97

TABLE OF CONTENTS (CONT'D)	<u>Page</u>
7.2 The Kernel Design.....	98
7.2.1 Design Model.....	99
7.2.2 User Interface Buffer.....	103
7.2.3 Program Management Interface.....	103
7.3 The Kernel Operation.....	104
7.3.1 Initialization.....	106
7.3.2 Daily Startup.....	111
7.3.3 Continuous Running.....	114
7.3.3.1 Changes to the Design Database.....	115
7.3.3.2 Answering Questions and Project Status Updates.....	117
7.3.3.3 Prospective Scenarios.....	121
7.3.3.4 Continuous Running Structure.....	123
7.3.4 Scenario Package.....	125
7.3.4.1 Prospective Scenario.....	126
7.3.4.2 Current Scenario.....	134
7.3.4.3 Cost and Risk Factors.....	138
7.4 Validation.....	140
7.4.1 Design Process Requirements.....	141
7.4.2 Management Requirements.....	145
7.4.2 Other Requirements.....	147
7.5 Summary.....	148

TABLE OF CONTENTS (CONT'D)	<u>Page</u>
8.0 SUMMARY.....	150
8.1 Review of the Parallel Model.....	152
8.2 Review of the Design.....	154
8.2.1 Performance.....	154
8.2.2 System Integrity.....	157
8.2.3 Items Missing From the Design.....	158
8.2 Conclusions.....	161
8.3 Extensions.....	163
GLOSSARY.....	165
LIST OF REFERENCES.....	169
BIBLIOGRAPHY.....	170
APPENDIX A.....	A-1
Kernel package.....	A-1
Design procedure.....	A-2
Data_Input procedure.....	A-2
Data_Input task.....	A-3
User_Interface_Buffer task.....	A-3
Program_Management_Interface task.....	A-4

TABLE OF CONTENTS (CONT'D)	<u>Page</u>
Design_Model package.....	A-5
Initialization procedure.....	A-6
Daily_Startup procedure.....	A-6
Continuous_Running procedure.....	A-6
Data_Input procedure.....	A-8
Allocator/Scheduler task.....	A-8
Database_Control task.....	A-11
Running Task.....	A-13
Scenario package.....	A-16
Prospective procedure.....	A-16
Current procedure.....	A-18
Stop procedure.....	A-19
Risk Factor procedure.....	A-20
Cost Factor procedure.....	A-21
Prospective task.....	A-21
Scenario_Database_Control task.....	A-25
APPENDIX B.....	B-1
Context Diagram.....	B-2
Data Flow Analysis.....	B-4
Structure Diagrams.....	B-6
Physical Design.....	B-8
Testing.....	B-10

LIST OF ILLUSTRATIONS

	<u>Page</u>
1.1 Management Expert System.....	7
1.2 Expert System Flow.....	9
1.3 Design Process.....	11
2.1 Design Hierarchy.....	15
2.2 Design Flow.....	16
2.3 Onion Skin Model.....	18
2.4 Data Flow Graph.....	20
3.1 Design Flow (modified).....	29
3.2 Design Flow (with parallelism).....	30
3.3 Design Flow (network flow).....	33
4.1 Resource Interleaving.....	46
4.2 Project Risk Factor.....	50
5.1 Project Risk Factor (with milestones).....	72
6.1 Management Expert System Use.....	77
6.2 Project's Current Scenario.....	82
6.3 Labelling of Tasks.....	86

LIST OF ILLUSTRATIONS (CONT'D)	<u>Page</u>
7.1 Kernel's Breakdown Structure.....	98
7.2 Kernel Structure.....	105
7.3 Data Input (initialization).....	107
7.4 Design Model (Initialization).....	108
7.5 Procedural Flow (Daily Startup).....	112
7.6 Design Model (Daily Startup).....	114
7.7 Procedural Flow (Change Generic Database).....	116
7.8 Procedural Flow (Answering Questions or Status).....	118
7.9 Procedural Flow (Prospective Scenario).....	122
7.10 Design Model (Continuous Running).....	124
7.11 Procedural Flow (Create New Prospective Scenario).....	128
7.12 Procedural Flow (Prospective From Current).....	131
7.13 Scenario Structure (Prospective).....	133
7.14 Procedural Flow (Current Scenario).....	136
7.15 Scenario Structure (Current).....	137

1.0 INTRODUCTION

To date, a number of System's Design Methodologies have been developed to somewhat formalize what has traditionally been viewed as a "Black Art".

The mechanisms used in a methodology have to be put down on paper (i.e., serially) in a way that the reader can follow the basic flow. These methodologies usually contain sets of activities separated by milestones which mark the completion of each set. Such is the case with the methodology developed by Bowen and Brown [1]. This methodology requires that all work on a particular phase of the design process be fully completed before proceeding to the next . This has the advantage that control points or milestones can be neatly placed at the end of each phase.

In real life, though, this presents a problem. Many times, except for trivial Designs, a stoppage can occur in a particular phase due to some unforeseen circumstance (e.g., not enough information is available to proceed on a particular module, or, due to a design error, a particular module does not work as expected, thus requiring re-design). This would cause the whole design process to wait until the anomaly is rectified. This results in delays in unrelated areas of the Design and resources that could otherwise be used, may not be available later. At other times, a company or its customer may wish to focus efforts on a particularly critical area of a Design and move it ahead while still

proceeding with the rest of the Design at a more reasonable pace. In this way, losses may be minimized in the event that the Design cannot be completed.

Because the real world operates in this type of "parallel" mode, serial methodologies cannot be efficiently used in their present form. The effect is that the natural control points which exist between the phases of a serial methodology, are lost. This requires a replacement control mechanism to be placed on the design process by the Project Manager which may not be suitable for the particular methodology being used.

To date, development of computer automated aids for design projects tend to be in two categories:

- design tools;
- and,
- program management tools.

Structured Design Tools help the Designer develop a product in a structured way. A Structured Design Tool could be one such as HP Teamwork/SA¹ which is used to create and maintain software specifications in a structured manner, or could be a CAD or CAE tool for helping in hardware design. Katz[4] says:

"Every type of project design - whether a printed circuit board, an entire system, or a VLSI chip - makes use of five applications of software: architectural specifications, detailed descriptions, verification, testing, and physical layout."

¹ A product of Hewlett Packard Company.

What Katz fails to mention is that even though structured design and other CAD/CAE tools help in the design of a Product, they do not necessarily help the Project Manager in the control of the design Project, itself. It seems that in order for a Product to be successful in the market today, the two major ingredients which must blend are:

- the simplest design which still meets the requirements [5 and 6] - simpler designs are less costly, easier to make, and get to the market more quickly;

and,

- good management of the project involving good communication amongst the players concerned [7 and 8] - good teamwork helps shorten the development time without sacrificing quality in the Product.

If either of these two ingredients is missing from the design organization, the Product will be more expensive and harder to build than is necessary or will take longer in getting to market, thus losing potential market share resulting in less profit. Design tools help in the first ingredient but not in the second. On the other hand, Program Management Tools (PMT's) help the Project Manager in organizing and controlling a Project.

Program Management Tools, such as SuperProject Plus² and Harvard Total Project Manager II,³ are typical of the aids that the Project Manager has today. These products help him/her in the control of a project in general but do

² A product of Computer Associates International, Inc.

³ A product of Software Publishing Corp.

not give enough help to a Manager of an engineering design Project. Some of the features that these products provide are:

- development of project organization using PERT;
- display and/or development using GANTT charts;
- resource loading and automatic levelling;
- costing of the Project;
- use of work breakdown structures;
- earned value analysis;

and,

- various methods of analysing and reporting status - including CSSR.

What these PMT's do not provide to the Project Manager of a design Project are:

- assessment of the risk involved in being able to complete the Project and create a producible Design;
- a guide to following a structured approach to design as it pertains to the Project itself;
- raise design and management issues in a timely manner as it relates to the particular Project;

and,

- the capability of tracking open issues assigned to the different tasks raised as a result of the above.

This can be a drawback because, in order for an available PMT to be used on a design Project, a lot of time and effort must be put into the overhead of molding and implementing the methodology around the restrictions of the Tool. This causes the Management Tool to become cumbersome to use. Due to these impracticalities, the extra effort may outweigh the usefulness of the methodology itself. Consequently, either the methodology or the PMT does not get used. This results in the lack of a proper blend of both good management and good design.

Since development tends to be focused on each of the above mentioned areas (and not on the combination of them), there is a requirement for a Management Tool that lends itself to both:

- a Structured Design methodology which closely models the parallelism in the work environment

and,

- maintaining the necessary control aspects of an effective design Project.

Therefore, the purpose of this thesis is twofold:

1. - to modify the existing methodology developed by Bowen and Brown (a "serial" methodology) into a suitable parallel model, thus allowing for maximum parallelism in the system's design process;

and,

2. - to create a Kernel of a Management Expert System (MES) which would be capable of handling such a parallel model along with the other management issues facing a project today.

The Kernel would not only control the flow of the design process as governed by the parallel model, but would guide the Project Manager (by means of questions and answers) through the various activities while highlighting potential problems to be addressed, tracking open issues, and determining the costs and risks involved with the Project plan. The Management Expert System (MES) would be comprised of the Kernel, a User Interface and some Program Management and other system Tools. This MES would be used as a control tool by a Project Manager or Project Engineer of a design Project. It would also be useful as a prediction tool during the generation of a design proposal and would help predict the potential bottlenecks and the risk areas before a contract is let or a bid is even given. One could do this because of the nature of the Kernel. The risk areas would be known because certain assumptions would have to be made in order to allow the Kernel to model the design process at such an early stage (all the User Requirements may not be known in complete detail).

Other than the Kernel, the Management Expert System will only be described in enough detail to illustrate the context in which the Kernel would operate. Because the existing methodology is intended for the design of embedded computer based systems, I intend to follow suit. Nevertheless, one

may wish to use the information contained herein for other types of design Projects.

1.1 The System

The Management Expert System (MES), itself, would be an expert system used by a Project Manager or Engineer in charge of a design Project. Ultimately, the system would be comprised of the Kernel, a User Interface, a Technology Database, other Program Management Tools and other System Tools. Figure 1.1 illustrates this in a form of an "onion skin" model.

Figure 1.1

The User Interface would consist of both menu driven commands and question and answer sessions. The menu driven commands would be used to access the Program Management Tools and Other System Tools. The question

and answer sessions would be used to access the Technology Database as well as to use the System's Design Model and other parts of the Kernel.

The Program Management Tools would consist of some sort of linear optimization program to generate data to be presented in the form of PERT or GANTT charts. The Kernel would also make use of the optimization program to help the Project Manager control the design Project.

Other System Tools would consist of a word processor, spreadsheet, CAE/CAD tools, access to the company's Local Area Network and other standard system tools normally found on a Project Manager's desktop computer. These would not all necessarily be used by the Kernel, itself, but would be available for the Project Manager if required.

The Technology Database would be a database set up for the use by the Kernel, the CAE/CAD tools, or the User . It could also be just a localized subset of a larger database used throughout the company. In this way, its controller could use the Local Area Network access mentioned in the previous paragraph.

Figure 1.2 illustrates how the User would view the system and how the different elements of the system would exchange data. In this thesis, the Program Management Tools, other System Tools or the Technology Database will not be discussed in detail. These could just as easily be covered after the Kernel and User Interface have been designed since these packages are already on the market today. Throughout the thesis, though, the interfaces from the Kernel to these tools will be discussed in order to show how the system would function.

Expert System Flow

Figure 1.2

1.2 The System Use

As in every process, there is a process flow. There are different levels of concurrency, precedence in the order of operations, critical paths and control points. The Project Manager encounters each during the design process. He/she would use the MES as a tool to exercise control over this process.

Although Figure 1.2 illustrates the architecture as viewed by the User, it does not indicate how, actually, to make use of such a system. As stated previously, the Project Manager would access the system through the User Interface. This interface would allow the User to access the Management Expert System via a Question and Answer session.

At first, the system would glean as much information about the design Project as necessary to create a design process flow. This would be illustrated in the form of either a PERT or GANTT chart. After the process flow is created, the Manager can then tailor the Project to his/her particular needs. Things affecting the Project would be time constraints, available resources (manpower or material), risk areas, effects of other Projects on the availability of resources and effects of those not directly related to the Project (but have some impact on the outcome of the design).

Using the process flow and the data input by the Project Manager, the MES would model the process and suggest control points, possible risk areas and the critical path of the Project. Because the MES has modelled the process based on the Project Manager's input, the Manager can then modify the criteria (such as manpower) to see what effects the change has on the flow of the Project. The risk areas, critical path and control points would then be dynamically changed based on the new needs of the design Project.

During the course of the Project, the Project Manager would input the progress and any other necessary information. Likewise, the MES would update the process flow to suit the current needs of the Project and suggest directions for the Manager to take based on the latest information. With this tool, the Project Manager could forecast his/her needs not only at the beginning of the Project but at any time throughout, and the MES would adjust accordingly.

The use of the system and the User Interface will be discussed in further detail in Chapter 6.

1.3 The Kernel

Figure 1.3 shows the model from which the thesis will start. Based on a set of Customer Requirements, the Logical and Physical Designs could operate somewhat in parallel and Management, Quality Assurance, Production and Test would work in the periphery.

Figure 1.3

What will fall out from this will be a flow model of the design process. From the flow model, one could determine:

- how much effort will be required for a design;
- how much will be required of Management, the Customer, Production and Test in the design process before the actual work begins;
- where the potential bottlenecks in the design Project are;

and,

- where the control points for the Project should be in order to avoid potential problems.

In Chapters 2 and 3, Bowen and Brown's methodology will be discussed and expanded into a model that would be suitable to base the Kernel on. This is necessary in order that the reader understand the basis on which to build. For the purpose of this thesis, it should be noted that the example chosen is considered to be typical of a serial methodology and that other similar methodologies may be suitable to the same expansion process. Chapter 4 will address other issues that affect the design but do not necessarily have a direct influence on the basic Project flow. References 5 to 9 cover some of these issues using case studies for illustration. Chapter 5 will take the issues facing a design Project and will show how the parallel model would handle them.

By Chapter 6, most of the design issues and the parallel model will have been covered in enough detail to start the design of the Kernel. In order to place the reader in the proper context before discussing the Kernel, the User Interface and some of the rest of the MES will be described.

In Chapter 7, the design of the Kernel will take place. Included will be Hierarchical and Procedural Flow diagrams as well as Structure Diagrams. Full Data Flow Graphs and Data Dictionaries will not be included in the Design because, for the purpose of this thesis, the Kernel's structure and its ability to handle the parallel model is more important than an analysis of the MES's processing speed or the flow of individual data.

Finally, in Chapter 8, there will be a discussion of how the Kernel handles the parallel model and the direction that one would take if the rest of the Management Expert System were to be designed.

Appendix A presents the Kernel in pseudo-code format. ADA⁴ will be used as the basis for the pseudo-code because of its ability to illustrate concurrent activities which happen in the design environment. This will be particularly useful from Chapters 3 and on since this is where the parallelism is developed.

Appendix B presents typical questions and status updates which are linked to form a logical flow through the different design phases of Bowen and Brown's methodology. These would form part of the Design Database for use by the Kernel of the MES. Because this Appendix deals directly with questions applicable to the sample methodology, it may not be suitable for other methodologies if they were to be used. Instead, a different set of questions, linked in a similar way, would be necessary as a substitute.

⁴ A trade mark of the U.S. Department of Defense.

2.0 THE BASIC METHODOLOGY

In this chapter, the methodology developed by Bowen and Brown will be covered briefly. This is a necessary point to start from because one must understand serially what must happen during the design process before he/she can address the parallel version. Since only the information that is required to start the design of the Kernel will be covered, if more background information is desired, it can be found in Reference 1.

The design of a system can be looked at from two perspectives: the logical; and the physical. Even though one may look at a product that already exists from the physical perspective, in order to approach a new design with an open mind, one must first start from the logical. Figure 2.1 shows the hierarchical layout of the whole design process whereas Figure 2.2 shows the basic flow. One may wish to refer to these figures while reading this chapter.

2.1 Logical Design

As mentioned, any new design must be first looked at from the logical perspective in order to achieve maximum flexibility during the design process. As can be seen in Figure 2.2, one must always start from a set of User Specifications. These must be in such a form so that a Designer can begin a design with enough information on the functionality of the end product. But there must not be so much information that the specifications cause an

extremely narrow path for the Designer to follow. This requires a very delicate balance on the part of the User (or Customer) so that he/she can receive the most cost effective design that still meets his/her needs.

Figure 2.1

To ensure that the correct information is transferred from the User to the Designer, a Context Diagram of the system is created. This Context Diagram is then compared to the Specifications before one can continue with the design. From the Context Diagram, a Data Flow Analysis is performed whereby the system is analyzed to decide where the data must flow in the system without much regard for how it will be transferred.

This process creates a Data Flow Graph and a Data Dictionary which are in turn a new set of specifications on which the Designer can build a structure. This structure is illustrated in the form of Structure Diagrams.

Design Flow

Figure 2.25

⁵ Taken from Figures 4.1 and 7.11 of Reference 1.

The Context Diagram, Data Flow Analysis and Structure Diagrams are discussed in greater detail in the following paragraphs.

2.1.1 Context Diagram

The Context Diagram, depending on the Designer, can take many forms. All that is necessary is that the proper information is written down by the Designer in an illustrative form so that both the User and the Designer understand each other. Bowen and Brown have chosen an "Onion Skin" model to illustrate the point.

To start with, the Designer must look at the User Specifications and separate the features into:

- functional requirements - these are requirements which are necessary for the proper performance of the system;

and,

- non-functional requirements - these are additional requirements imposed by the Customer which would be the first to be changed if the system becomes not technically practical (e.g., processor to be used or language the software is to be written in);

and then,

- User visible functions - these are functions which would be in the User Interface or may be just visible by the User;

and,

- the interactions between the different sections of the system as may be defined by the Customer.

The above features would be tabulated in the Designer's Database. If there were noticeable functions missing from the database or if there were any questions with regards to the functioning of the system, the Designer would have to consult with the Customer in order to resolve the outstanding issues.

Figure 2.3

From the Designer's Database the Context Diagram is created. The "Onion Skin" model (Figure 2.3) has two types of partitions, concentric and radial.

The concentric partitions separate the system into different layers of a virtual machine. As one gets closer to the center of the model, the functions are carried out by the heart of the system. Radial partitions, on the other hand,

separate the functions into groups called Modes. These are separately functioning entities which only exchange data via a specified interface.

2.1.2 Data Flow Analysis

Once the Context Diagram is complete, the Designer can start doing a Data Flow Analysis. This is an operation whereby the Designer starts sizing up the system to be designed and starts to develop efficiencies in the flow of the system. The functional elements and some of the necessary interfaces have been identified in the previous operation. Now, the necessary data to be transferred along with the processing functions have to be identified. During this analysis, a Data Flow Graph and a Data Dictionary are created.

After the Data Flow Graph and Data Dictionary have been created, the maximum data flow rates and the minimum execution times possible for the system have to be determined. These rates must then be compared to the User Specifications and then the system must be modified to meet the Specifications at minimum cost. This may mean further subdividing the functions to increase efficiency or coming to a compromise between the User and the Designer.

After the User Specifications have been met, the system can be reduced to achieve minimum cost. This is done by slowing down required transfer rates and data transformations or by reducing parallelism without reducing the overall system performance. Overall system performance must not be reduced at this time because the control overheads are not yet known.

2.1.2.1 Data Flow Graph

With the Designer having the proper context of the system that is required by the Customer, the Data Flow Graph is created. This is a set of nodes (processing functions) and edges (data paths) placed in a format in which the data flow can be analyzed. As a standard, data flows only in one direction along any particular edge. The overall flow in the system would be from left to right and from top to bottom. Of course, except for a trivial system, it is not possible for all edges to adhere strictly to the standard. Some functions may have to be divided in order to keep the "left to right" rule in tact.

Figure 2.4

Once the top level diagram is in place, then it can be subdivided first in a pipeline fashion and then in a parallel manner until there is sufficient granularity for the Designer to determine the minimum execution times and data flow rates

possible. Figure 2 4 illustrates a simple Data Flow Graph. Of course, the node names and processing rates, and edge data flow rates would normally also be included when the graph had sufficient granularity.

2.1.2.2 Data Dictionary

The Data Dictionary is a database set up by the Designer for the system being designed. It consists of a list of all the nodes in the system along with their data processing rates and a list of all the edges in the system with all their data flow rates.

2.1.3 Structure Diagrams

At this point, all the preparation has been carried out and the design is ready for the structure and control to be placed on it. This procedure constitutes probably the largest part of the logical design process. Now, if the Data Flow Graph could be looked at from left to right, the Structure Diagrams can be viewed as the control coming from the top, down onto the data.

As with the Data Flow Graph, the Structure Diagram starts with a top-down approach to design. The whole process can be looked at in two parts at each level of design:

- 1 - establishing the actual control structure of the system as follows:

- a. - determine the mutually exclusive modes, functions and input/output data resources - the modes and functions would be tasks and the resources would be represented as passive packages;

b. - group the modes and functions into packages;

c. - using the following guidelines, create the actual Structure Diagram:

i - target task (the one receiving data), low utilization - configure target as a server,

ii - target task, high utilization - configure target as an actor,

iii - many senders, either target task busy or many targets - use buffer task as a server,

iv - passive packages or limited resources (e.g., data bus, memory, disk, etc.) - may require an allocator or arbiter task;

d. - proceed to the next lower level until the desired level of granularity is desired;

and,

2 - determining the extra timing and data necessary for control and update the Loading Analysis and Data Dictionary accordingly.

Although some may argue that an actual Structure Diagram is not necessary to continue with the design, few will disagree that the control issues in the design, themselves, must be addressed. The Structure Diagram is just a medium for the Designer to illustrate the logical control within the system. It

would be meaningless to illustrate a Structure Diagram at this point without an actual example of a design. Examples of these diagrams and their development can be found in Chapter 7 during the design of the Kernel.

2.2 Physical Design

Once the logical part of the design is complete, it is necessary to start the physical. But, there still remains some preparation before the actual incorporation takes place. This preparation involves an investigation into the present technology and existing designs to see what can be borrowed for the new system. New technology may be available but not yet fully developed to the point of a working design and still newer ideas may have to be developed specifically for the new system. Of course, the more recent the technology, the riskier the design of the system will be. This must then be weighed against the older more "tried and true" designs for the actual added benefit it provides for the system.

2.2.1 Preparation

The preparation involves both technology at present and development of newer ideas that have not yet been implemented. Both of these avenues must be explored before a final approach to the design can be decided upon.

2.2.1.1 Technology

Reviewing the current technology may not be a very long process if the Designer is already familiar with what is already on the market. But, a new design usually entails some aspect that relies on new ideas and new ways of

using current techniques. The decision of what processor to use or what scale of integration to use in silicon depends on the capabilities of the technology, the Designer and the support that is readily available. It serves no purpose to decide on a microcircuit or other device for a design which will not be available until after production is due to finish (even if the part is ready part way through production, provisions will have to be made to account for the shortage). At each level of the design, the current technology must be reviewed in order to not "re-invent the wheel".

Software is also dependent upon the resources that are available during the design process. The type of software chosen is dependent upon the processor chosen, development tools available and the application. Some languages are not suited for certain types of applications or may require cross-compilers which can create inefficiencies in the design.

2.2.1.2 Partitioning and Allocation

Now that there is a good understanding of the technology available, it is time to stand back and look at the Logical Design of the system and start partitioning (dividing it up into physical "chunks") and allocating the functions to hardware and software.

Partitioning can be either of a parallel or pipeline nature. The alternatives can take several approaches depending on the design's priority on the following:⁶

- partitioning with respect to the inherent parallel and precedence relationships of the system;
- balanced data flow rates;
- balanced computational loads;
- system modularity and expandability;
- minimum system memory requirements;

and,

- predefined implementation hardware capabilities.

After the system has been partitioned, the different modules can be assigned to hardware (or software) and then a further loading analysis must be performed to verify that the design still meets the original User Requirements.

2.2.2 Incorporation

As in Logical Design, this process should begin from the top and proceed down to the lowest level of the design. This creates a highly modular product which is easier to develop, debug and maintain at a later date.

⁶ Taken from page 208 of Reference 1.

The incorporation of the design involves the bulk of the whole design process. Although this can be a very large task and seems to have a somewhat lesser place in this text, the job and risk of the incorporating process can be greatly reduced if a lot of work went into the previous sections. The act of incorporating the design from the preparation, both the logical and physical, involves creating drawings, making prototypes or breadboard models of the riskier designs, developing software, designing custom chips, developing the mechanical enclosures and other related parts and making the design ready for production.

The incorporation should be, again, of a top-down nature, not only because of modularity but the higher levels should be the first parts of the design which would be ready for incorporation. This helps in the next chapter when parallelism is introduced into the design process.

2.3 Summary

In this Chapter, the basic design process flow was discussed. This involved the transformation of a Design from the User Requirements, to the Designer's Context Diagram where the Designer acquired as much information as he/she could from the User. From there, the design had a Data Flow Analysis performed on it and control structure placed on it in the form of a Structure Diagram. From the Structure Diagram came the transformation to an actual Physical Design in both hardware and software. The Physical Design was further refined and compared to the original User Specifications until the whole Design, both Logically and Physically, met with what the User required.

3.0 THE METHODOLOGY WITH PARALLELISM

Now that the basic methodology has been described, it is necessary to create the actual flow of the design process. As was shown in Figure 2.2, the design process basically flows in a positive pipeline-type direction from the initial User Specifications to a final Design. Of course, except for the most trivial cases, there is always redesign and instances where the Designer has to re-think what is to be done. This disrupts the basic process flow by causing either a stoppage of the whole process until the error is corrected or a deviation from the strict adherence to the precedence relationship of the flow. This must all happen with maximum throughput of design effort. Some areas of the Design may be labour intensive, thus taking longer to complete than the rest. This causes a wait period or a bottleneck in the Project.

Therefore, one can assume that in order to have maximum parallelism, the design process must not be bogged down due to a single design event or a lack of a single resource. It should be able to adapt dynamically to the situation at hand and still allow the Design to continue (albeit less efficiently than with an ideal process where there are no flaws). Of course, there comes a time when some issues have to be cleared before anything can proceed. On the other hand, if the process gets to this point, it would be in serious trouble anyway (e.g., the Design is not technically feasible). Therefore, it will be assumed that graceful degradation is key to an efficient design flow.

In this chapter, the basic methodology, as presented in Figure 2.2, will be split in order to create some parallelism but it will still maintain some of the pipeline nature it already has. Then, it will be shaped into the form of a network where there will be few distinct points at which the Design must wait until a particular phase has completed (i.e., all the Data Flow Graphs being complete before continuing with the Structure Diagrams). Then greater efficiencies in the process will be discussed. Once efficient parallelism is achieved in the design process, one can then create the necessary interfaces to the other issues that have an effect.

3.1 The Flow of the Design

Figure 2.2, which is basically a pipeline, can now be expanded as shown in Figure 3.1 to create the parallelism as shown in Figure 3.2.

Once the Design's features have been tabulated, the process can start being split up. It is not recommended that the Tabulation task be split because this is where the Design begins and all future work is based on its outcome. If a mistake is made at this stage, it may be carried throughout the Design causing extra work and possible schedule slippage. As can be seen, in each of the stages following the tabulation, the process can be further subdivided to accommodate greater parallelism. The procedure for verifying that the Design meets the requirements, at the bottom of Figure 3.1, should be viewed as a building process, bringing the whole Design together as a system, rather than as a dividing process. Then, at the end, the whole system can be compared to

the original User Specifications. If some part of the Design does not meet the Specifications, only that part (or parts directly affected) need be redesigned.

Figure 3.1

Figure 3.2

The idea of a structured system brings to mind the word *modularity*. The methodology, as presented by Bowen and Brown, creates a modular system in that there are distinct modes, submodes, etc. Each is developed into Data Flow Graphs with nodes and edges, then into Structure Diagrams and then is

partitioned in the physical part of the Design. There is an established mechanism by which individual functions are traceable back to the User Requirements. This process of maintaining the system in a modular or hierarchically structured form will be defined as modular consistency.

The process of maintaining modular consistency will be extended to involve not only dividing and combining along module boundaries, but also include the idea of keeping work assigned on a particular module consistent with its boundaries. If, during a process (e.g., creating a Context Diagram), work on a module is split into two sub-modules without first creating the proper interfaces, there is a chance that data will be lost and the functions established in the User Requirements could be corrupted.

For example, if a satellite system were being designed, the overall Context Diagram could be sub-divided into two separate parts: the ground system and the satellite itself. Of course, the control and communications links would have to be established, but each part could be worked on by separate groups. On the other hand, one and a half Context Diagrams in this example would tend to create chaos because of fuzzy definitions of the interfaces involved.

Therefore, modular consistency should be a necessary part of the design process in order to lesson errors and for the realization of parallelism in the work environment. Otherwise, the problems caused by not having this consistency may override any benefit that may be realized. Errors caused by omissions end up with delays to the program due to rework. With the added work comes added cost.

following chapters). Further design of the receiver and transmitter need not be interrupted because they are not affected by the change to the display.

3.2.4 False Technical Assumptions

In some cases, at the beginning of a design Project, not all the facts of the Design are understood. When this is the case, some technical assumptions are made by the Designer. Later on in the design process, the Designer is in a better position to modify the assumptions or even to fully eliminate them. At this point, he/she may realize that the original assumptions were wrong and there are errors in the Design.

In order to lessen the propagation of errors throughout the system, assumptions should be reviewed before moving on to a following stage in the design process. For example, if there were an assumption originally made during the creation of the Context Diagrams because technical information was not currently available, say, and then between the Data Flow Graph and Structure Diagram stages it were able to be "closed out" by investigating the problem more deeply, the chance of its being forgotten about later on in the Design would be lessened.

Nevertheless, errors usually do surface in a Design and they should be corrected as soon as is feasible in order that they do not propagate further. The correction process would be generally the same as if there were a change to the User Requirements and would follow that of 3.2.3 above.

Figure 3.3

and,

4. - If necessary to combine modules, only build the new module with an integral number of smaller modules - leaving parts of modules behind, again creates a chance for functions to be forgotten and the system to be corrupted.

Although the above rules do not eliminate the possibility of the design Project's going astray, they do help to lessen the chances because the functions are traceable back to the original User Requirements.

At this point, each part of the design process can be regarded as an operation having a set of antecedents, a set of consequences and taking a specified period of time (Figure 3.2). This greatly simplifies the way in which the complex and diverse aspects of the design Project may be viewed. From Figure 3.2, one can then expand the view of the process into one as shown in Figure 3.3. As can be seen, the process flow turns into something resembling a Network Flow Graph. Because each part has a required time attached to it, the network can then be used as the basis for schedules, resource loading requirements, and completion date estimates. In this way, the whole design process can be treated as a linear programming model and can be mapped onto a standard program planning package, which would illustrate the flow of the Project in a way that suits the Project Manager. The network aspect of the process has another advantage - critical paths can be established. Once a critical path in the design process is recognized, the Project Manager can then concentrate his/her efforts on shortening the path (if possible and/or necessary).

This is so that problems on the critical path can be paid attention to first, thus lessening the chance of slipping schedule.

3.2 Process Flow Issues

The process flow issues of the Design will be discussed in a number of categories:

- stoppages, workarounds;
- resources - manpower, computer/instrumentation;
- change requirements/design;
- false technical assumptions;

and,

- convergence on a Design.

3.2.1 Stoppages

Stoppages can be treated in either of two ways: stop the process and correct the cause of the stoppage; or, adjust the process to accommodate the cause of the stoppage until it is corrected.

Stopping the process in the particular spot until the stoppage is corrected would probably be the cleanest of the two approaches because there is no deviation from the basic original plan (just an addition to the plan to correct the problem) and added costs due to the stoppage would be lessened. This would be particularly useful if there were slack in the schedule or the impact of the stoppage would not affect the critical path and if all the resources would be available when needed. The disadvantage to this approach is that the Project

Manager may like to save the slack in the schedule for a time in which he/she feels is a riskier area of the Design.

The latter approach of adjusting the process to accommodate the stoppage, although causing some headaches during the adjustment, allows for most of the process to continue and to utilize the resources which may not be available later. This has added cost because the process would be less efficient than if the problem did not exist. It also adds to the time in which the whole process is carried out due to inefficiencies. This loss of time, though, may be regained somewhat at a later date because the resources were used when they were available, thus allowing for greater efficiencies elsewhere in the design process.

Neither of these approaches suits every situation ideally. Therefore, the Management Expert System could be used to try each scenario before implementation to help the Project Manager decide which would best suit his/her particular needs and priorities.

3.2.2 Resources

The resources that are available for a Project, both manpower and instrumentation, can govern the flow of the design process. Of course, the Project Manager has input to what is currently available for the Project but sometimes, unexpectedly, they may be removed. This can happen by any of the following:

- the equipment breaks down and replacement parts are not available;

- an employee on the Project quits or is taken off the Project for another higher priority one;

- equipment or material is late in arriving at the facility;

- equipment is allocated to another Project;

- another part of the Project has problems and requires more attention;

or,

- another Project has slipped schedule and the resources are not yet freed for use.

On the other hand, the Project, itself, may require more resources than originally anticipated due to having to regain lost time or simply, that the Project was larger than expected.

Nevertheless, whether resources are taken away or the fact that more are required than are currently available, the process flow will be affected. The Project Manager must decide on whether to live with the new flow path or to modify it by acquiring more resources. Chapter 4 will cover the management issues relating to the addition of resources. For now, it will suffice to say that the Management Expert System should help the Project Manager by showing the potential effects each scenario will have on the Project before actually incorporating them.

3.2.3 Changes in Requirements

Changes in Requirements or Design, either generated by the Customer or by the Company can cause a disruption in the design flow. The decision on whether the changes should be made or not is a management issue and will be covered in the next chapter.

If there is a change in the requirements of the Design, the change must be examined to see what part of the design process (i.e., Context Diagram, Data Flow Graph, etc.) will have to be repeated or revised. Also, if the rest of the Design is to proceed, the possible future effects to the design process should be analysed. Then, the necessary parts will have to be repeated for that portion of the Design which is affected by the change. It is recommended that the part of the Design, which is directly downstream from the point in time when the change was realized, should be placed on hold until the change is incorporated into the Design. This limits the further errors that could be created in the Design by not incorporating the change. Although, if schedule does not allow for the Design directly downstream to stop, it would be a management decision to proceed at risk.

Say, for example, a radar system were being designed. Basically, it would consist of a transmitter, a receiver, a processor and a display. If for instance, a new feature was desired of the display to make it more usable, then any further design of the display should be halted until all the re-design is complete. Depending on the demand the change would make on the processor, it may or may not be halted. If it were to proceed it would be at the risk of having to re-design it later on (risks will be discussed in more detail in the

following chapters). Further design of the receiver and transmitter need not be interrupted because they are not affected by the change to the display.

3.2.4 False Technical Assumptions

In some cases, at the beginning of a design Project, not all the facts of the Design are understood. When this is the case, some technical assumptions are made by the Designer. Later on in the design process, the Designer is in a better position to modify the assumptions or even to fully eliminate them. At this point, he/she may realize that the original assumptions were wrong and there are errors in the Design.

In order to lessen the propagation of errors throughout the system, assumptions should be reviewed before moving on to a following stage in the design process. For example, if there were an assumption originally made during the creation of the Context Diagrams because technical information was not currently available, say, and then between the Data Flow Graph and Structure Diagram stages it were able to be "closed out" by investigating the problem more deeply, the chance of its being forgotten about later on in the Design would be lessened.

Nevertheless, errors usually do surface in a Design and they should be corrected as soon as is feasible in order that they do not propagate further. The correction process would be generally the same as if there were a change to the User Requirements and would follow that of 3.2.3 above.

3.2.5 Convergence

Since the aim of a design Project is to create a product from a set of User Requirements, it is necessary to discuss the issue of convergence. Given that the parallel design model does converge into a Design, the issue in this thesis remains with the speed or rate of convergence. Does the added parallelism add or take away from the rate of convergence? One would think that if the job were divided up, the design process would proceed more quickly. But, taken to an extreme, if the added parallelism just confused the issue and created more errors and problems than were corrected, then the design process would never end.

Therefore, the Management Expert System must aid the Project Manager in reducing the unknowns and help him/her track the open issues and assumptions so that they get resolved before the Design goes astray. The MES should help the Project Manager by flagging when the open issues become important, when they start affecting the critical path, and what will potentially happen to the Design if the issues are not addressed right away.

3.3 Summary

In this chapter, parallelism in the design process was addressed and other process flow issues were raised.

With the parallelism, the whole idea of modular consistency, not only in the Design but also in the process of creating the Design, was brought up. This is necessary in order to reduce the chance for errors in the Design as well as for

the proper transfer of information between design activities. The idea of the design activities within the process flow having a standard structure (antecedents, consequences, and time) allowed all the activities to be regarded in the same light. This allowed the process to be mapped onto a Network Flow Graph, thus allowing it to be manipulated using PERT or other management techniques.

The questions of stoppages, resources, changes in requirements, and false technical assumptions were brought up. The Management Expert System must address these in order to allow for the convergence of the Design.

4.0 MANAGEMENT AND OTHER ISSUES FACING A DESIGN

This chapter is going to continue with management and other issues which have an effect on the design process. These effects are not all necessarily a benefit to the design process but it is hoped that most of the bottlenecks that one could encounter during the process will be covered (It would be foolish to hope to cover all of the issues because in any process *Murphy's Law* always reigns supreme).

The management issues have been divided into the following categories, each of which covers some part of the management aspect of the design process:

- Economics of the Design;
- Allocation of Resources;
- Control Issues;

and,

- The Role of the Customer.

Other issues which also have an effect on the design process are:

- Quality Assurance;
- Production and Test;
- Documentation;

and,

Logistics Support.

These aspects and how they affect the design process will be discussed as well. Although some would argue that they are not necessarily directly involved with the Design, they could quickly bog down a Project if the Design ever had to be sold as a Product.

Issues raised in each of the following sections may not be directly addressed by the Management Expert System but could be simply highlighted as reminders to the Project Manager at different parts of the design process.

4.1 Economics of the Design

The ability of the Project Manager to accurately predict costs of the design Project depends highly on his/her experience and his/her ability to relate past experience to the current Project. This is not to say that other people do not have a part, but that the Project Manager is responsible for the Project and should be confident in his/her predictions. The Management Expert System should help the Project Manager in predicting costs by showing the planned design process flow and the potential risk areas. Because the design process is carried out in a structured manner and the MES shows the predicted paths to the end of the Project, the Project Manager can forecast costs of each node in the flow, each with a confidence level assigned. Therefore, he/she will have a greater confidence level in the total cost of the Project. Also, as a database is built, based on actual costs of past Projects (or past sections of the current Project), the MES will further help the Project Manager in his/her estimates.

Another way of looking at the "Economics of the Design" is to predict what the cost of the end product would be after it is produced. If the product is too costly for the Customer, a competitor's may be less expensive. Consequently, it may not be possible to sell the product in quantities as originally expected. Therefore, it is important to keep the cost of the end product down without compromising the User Specifications.

Changes in requirements usually affect the cost of the product in some way. Some questions that should be asked before the change is incorporated into the Design are:

- how is the change going to affect the cost of the product;
- and,
- who is going to pay for the change?

The Customer will have to pay for the change either up front during the design phase or in the end price of the product. The overall price of the product to the customer would be the cost of designing the product, the cost of producing it (including warranty provisions, etc.) plus the profit the company seeks to obtain. The change could lower the end cost of producing the product thus increasing the profits of the Company (in this case the Company may pay for the change).

Another aspect for consideration is compatibility. In order for a product to be compatible with others in the future, special considerations should be made during the design phase. These would be added requirements imposed by management increasing the cost of the present product with consideration of the savings that could be realized with future versions. Management may have

to live with the extra cost in order to remain competitive. On the other hand, the customer may pay for the extra features knowing full well that the product can be expanded upon at a later date.

In light of the above, tracking and predicting both types of cost during each phase of the Design are important. Therefore, some mechanism to allow this should be part of the MES.

4.2 Allocation of Resources

As mentioned in the previous chapter, the Project Manager has an influence over the resources available but he/she does not have complete control over them. Due to unforeseen circumstances they may be suddenly taken away. Sometimes, the resources the Project Manager has do not directly suit the task required. For example, a logic analyzer is required but only an oscilloscope is available or, in the case of personnel, a hardware designer with some software experience is available where a person with a more extensive software background is required. In both cases, there is a tradeoff as to whether one should wait for the proper resource and possibly slip schedule or to use what is available in a less efficient manner. Slipping schedule too much does not look well in the eyes of the customer causing the Company to lose future sales. Making do with the available resource costs more thus reducing the profits or increasing the end price of the product. In the case of personnel, there may be a learning period where the person not ideally suited for the task has to develop new skills. When a new person is brought onto the Project, there is again a learning period for him/her to learn about the Project.

The parallel model keeps the Design modular in order that a new person would only need to learn the parts that directly affect him/her with a general overview of the whole product. The model lifts other restrictions and allows operations to be staggered in order that a resource can be allocated to tasks not happening at the same time. This lessens the backlog of tasks being delayed while waiting for the use of the resource or need for duplicates to alleviate the problem. If tasks operating in parallel can interleave the use of resources enough (Figure 4.1), ideally only one of each resource would be required.

Note: Boxes are tasks,
Letters are resources.

Figure 4.1

Therefore, one can assume that except for special circumstances, the resources the Project Manager does have are limited. Consequently, he/she must make maximum use of the resources at hand and plan for the resources needed in the future. More importantly, the Company as a whole must maximize the use of the resources it has and allocate them for the maximum benefit of the Company. The Project Managers of different Projects should be

able to use the MES to develop scenarios and prepare a cost/benefit analysis on each so that management can decide how best to meet the needs (both time and budget) of all the Projects.

4.3 Control Issues

Probably, one of the major problems with parallel operations is the control aspect. With a pipeline type of methodology (as in Chapter 2), there is not as much of a problem because there are distinct points where one could exercise control (i.e., between the Data Flow Analysis phase and the Structure Diagrams phase). In the parallel model, with different tasks at different stages of design (as in Chapter 3), and different people conducting them, how does one assert control over the whole process? When the schedule indicates where the Project is, how does one know it is accurate (and not further ahead or behind than is indicated)? When tracking costs, how does one know that the work performed equals the progress made (not 90% of the money spent on 10% of the work required)? How does one ensure that the same new ideas are not developed by two different Designers? In order to save costs, new ideas should be developed once and then borrowed for other applications.

In order for the MES to be useful as a tool for a Project Manager, it should address the above questions. Even though the design process is parallel in nature, there is progression along each path where one proceeds from the Context Diagram through to the Physical Design. Each path of the Design follows a similar process at almost the same time and in the same order.

Therefore, one could choose points at various times in the process and label them as Control Points or milestones.

Looking at Figure 3.3, one can see that even though the individual phases (e.g., Loading Analysis) can be staggered between tasks, there is a time when all similar phases in the Design are complete. These points could be used as Control Points. Recommended Control Points would be at the end of the Context Diagrams, Loading Analyses, Structure Diagrams, Physical Designs, and the last test for meeting the User Requirements.

J.G. Cottrell states:⁷

"Stage by stage control must also ensure that:

- (1) Stages are not allowed to overlap.
- (2) Shortcomings against the objectives at any stage are cured before the next is initiated.
- (3) Approval of the results of each stage involve commercial and technical functions and with general management involvement in the final decision making go/no-go process.
- (4) Programs do not proceed without full engineer manning since otherwise the engineering will be inadequately carried out particularly if tight time-scales are involved.
- (5) Changes to specification may only be introduced by management decision and with full prior adjustment of time and money, where necessary, to the programme."

Therefore, one method of control would be to review the Design up to the Control Point. If satisfactory, incorporate a "Design freeze", whereby after the freeze another formal review would have to be conducted in order to change anything already frozen. These formal reviews for changes should look at how the changes would affect the rest of the Design in the areas of User

⁷ Taken from page 229 of Reference 3

Requirements, cost (both of the redesign and of the end product), the necessary redesign, and schedule.

Ideally, one would like to have 100% of a phase complete before a review and a "Design freeze". In this respect, Cottrell's second statement is agreed with, but this is not always practical. If one part of the design were extremely late or had to be redone before being accepted, in order to complete the review a lot of resources would be tied up waiting for the unfinished part. Therefore, it is recommended that the MES allow the Project Manager to choose a level of completion for a certain phase (say 90% to 95%) before conducting a review at the Control Point. The control process would then proceed as follows:

1. - During the review, all parts of the Design would be discussed including the parts not fully completed;

2. - A list of open items would be established including discrepancies in the Design and the tasks not fully completed;

3. - Any items not on the list of open items would be "frozen" and those parts would proceed "at risk" until all the opened items are dealt with - Attached to the list would be a schedule with completion dates;

and then,

4. - Following the last date on the list, a mini-review would be conducted in order to close off the open items, weigh the impact on the

rest of the Design already proceeding and "freeze" the Design up to that stage.

This allows the Design to be reviewed without causing undue delay and frees valuable resources to carry on with the Design.

This brings to light the idea of a "risk factor" in a Design. At any time during a design Project, there is an element of risk that the path on which the Design is proceeding is wrong and there must be redesign. At the beginning, the risk is high (say 100%) since there still remains a lot to be done. Hopefully, at the end, the risk is low (say 0 to 1%) since if the Customer has accepted it, the Design should meet the User Requirements. Anytime in between, the risk should be decreasing since progress is being made, reviews are being held and Control Points are being passed.

Figure 4.2

Figure 4.2 illustrates risks associated with a set of paths on which a design Project can follow. The desirable path is one which the risk is greatly reduced at the beginning of the Project. It is desirable for many reasons, two of which are cost and schedule. If risk is low, the chance of redesign is low. Therefore, cost will be low and the chance of meeting the schedule is greater. On the other hand, if risk is high near the end of the Project, there is a good chance that there will be redesign along with added cost and schedule slippage. This is therefore an undesirable path to take.

A number of factors can increase the risk on a design Project, some of which are:

- technical assumptions without valid backup;
- open items left for a later date after a major review;
- unanswered questions to the Customer;
- exceptions to a structured approach to design (skipping steps);
- lack of modular consistency;

and,

- changes in a part of a Design which was already "frozen".

Since the above factors increase the risk in a Design, it would be in the best interest of the Project Manager to eliminate as many of the above as is practical. Sometimes it is far too costly to eliminate an item at a particular time. The Project Manager would therefore weigh the costs of closing an item versus the risk of leaving it open, and choose which best suits the Project while keeping in mind Figure 4.2. The MES should help the Project Manager in making his/her decision by showing the related risks and costs of reducing them.

The above Control Points ensure that ground already covered does not get retraced unless a conscious decision is made by the Project Manager. But that still leaves the questions, "How does the Project Manager really know where the Project actually stands (with respect to both schedule and cost)?" and, "How does one ensure that work does not get duplicated in two parallel operations (particularly in the lower levels of the design hierarchy)?"

Following the design process as outlined and properly completing each phase before moving on to the next will provide structure to the process. After acquiring some experience, the Project Manager will have a good feel for where the Project stands. This does not guarantee that there will not be redesign due to improper technical assumptions or mistakes in transferring the Design from one stage to the next. The MES will help matters due to the consistency and the strictness one would have to follow during the design process. History could be recorded in a database and after a few entries, the MES would help the Project Manager by recalling related past experiences.

It can be argued that duplicate operations or Designs are a different management problem. One can only tell if a function is related to another (directly or indirectly) by actually examining the design specifications at the particular phase in the Project. Since the MES would only help in the management control of the process without actually looking at the Design, this problem is beyond the scope of this system. One could, however, include a question to be asked before each task is started requesting if the following work has been completed in another area of the Design. This would remind the

Project Manager to review the Design to weed out similarities and reduce unnecessary work.

4.4 The Role of the Customer

Although the Customer usually plays a relatively small role during the design process, he/she does have a major impact on the outcome of the Design and should have a much larger one. The reason for this is that the Customer must provide a complete set of User Requirements. If the requirements are not well defined at the beginning of the design process, the Design could very well turn out to be what was not needed or hoped for. At the beginning of the Project, as well as at the various Control Points throughout, the Customer should be aware of how the Design is progressing. If the Design is going astray and is not meeting the original requirements, the Customer should make it known so the problems can be rectified before too much time and effort is wasted. For example, in a fixed price contract, if the errors were a result of the Company, then the only loss to the Customer would be with respect to the time that the product would be delivered. But if the errors were a result of incorrect User Requirements that were not clarified, then the Customer may be faced with a Change of Scope to the original contract or a product with too many features resulting in its being more costly than it should be. Either way, it is in the Customer's best interest to answer any questions and clear up any problems as quickly and as early in the design process as possible.

Sometimes during a Project, a problem which was not covered in the User Specifications is encountered. In this case, it is also in the Customer's

best interest to confront the problem and ensure that the Designer knows what is desired. Approvals of changes required or for deviations from the User Requirements should be followed through in order not to delay the Project. At the end of the Project, it is the Customer's responsibility to accept the design if proven to meet the User Requirements (including any approved changes or deviations) and the requirements of the contract. This is a necessary step to complete the design process.

4.5 Quality Assurance

The Quality Assurance (Q.A.) department of the Company can play a key role in the design process. It goes without saying that if one wants to survive in today's marketplace, one has to produce a quality Product. Some people talk about the "different levels of quality" in Products. What this really means is "to what extent is one willing to pay to be assured that the Product is going to carry on doing what it was originally intended to do?" The Canadian Department of National Defence and the United States Department of Defense publish standards of quality that they require in the Products they buy. The National Aeronautics and Space Administration (NASA) similarly produce specifications for the Products they buy. Other companies and organizations also produce standards for commercial or industrial use or simply adopt the same as the military or NASA. Nevertheless, the Q.A. department's job is to ensure that the Customer is going to receive the Product he/she is expecting to receive.

In the User Requirements, there is usually a section which contains the standards of quality that the Customer wishes to see in various aspects of the Design. These could include:

- drawing standards;
 - software design standards;
 - configuration control of the Design;
 - quality of parts to be used - level of burn-in and type of screening required;
 - manufacturing standards - these would have to be incorporated in the drawings;
 - any environmental or other performance specifications;
- and,
- standards for testing and proving the Product.

The Q.A. department ensures conformance to these standards. It does this by reviewing the Design at various stages throughout the design process and by witnessing the qualification or acceptance testing of the end Product.

If the Q.A. department is consulted during the design process, it can be kept informed of the progress of the Design as well as give suggestions in order to save time and energy. The experience the Q.A. people have in staying within the standards of both the Company and the Customer can prove to be

invaluable. The MES should make reference to consulting the Q.A. department at various stages during the design process (possibly at the control points).

4.6 Production / Test

The producibility of a Product affects the end cost which in turn affects its price and its attractiveness to the Customer. If it is very hard to produce a Product, the chance of damage and therefore rework will be high. On the other hand, if it is relatively easy to produce, the amount of scrap will be lower and therefore costs will be lower. Sometimes new techniques will be required which will take time for training, but as long as they are repeatable, they can be acquired and a quality Product can be produced consistently.

When considering the producibility of a Product, one should also pay attention to where the different parts are going to be produced. The Company has production facilities which have certain manufacturing strengths and weaknesses. When each part is being physically designed, one should pay attention to where the part is going to be produced: in-house or not.

If it is going to be produced in-house, one should keep in mind the equipment that the Company currently has. If it is not adequate, then there has to be a management decision as to whether to invest in the new equipment, to re-design the part in order for it to be produced in-house or to buy the part from another plant. It can be seen that if the Company wishes to produce the part in-house and it is not currently capable, then there will be costs incurred.

Alternatively, the part can be produced outside the Company. This does not relinquish the Designer's responsibility to understand the capabilities of the other manufacturers. This is due to the fact that all the costs incurred will be passed back to the Company in the price of the part. Sometimes, in order to keep costs low, it helps to consult the other manufacturers as to the best ways to match the demands of producing the part with their production capabilities.

Once the Designer knows the part is capable of being produced at a reasonable price, then come the production efficiency considerations. There are usually cost tradeoffs depending on how many parts of a particular type are being produced. Sometimes, if there are many identical parts, the up front costs of tooling or setup for a more efficient process can be amortized over all the parts to be produced. If there is a more efficient way of producing the part, costs per unit could be lower. Again, the considerations of where the part will be produced (in-house or not) should be taken. Efficiencies in one manufacturing facility are not necessarily true for other facilities.

Ideally, if the Design meets all the User Requirements and there were no errors in the drawings and no manufacturing errors, one would not have to consider rework. Unfortunately, this is usually not the case. Rework has an impact on both costs and schedules. Damage due to rework compounds the problem. Therefore, if possible, allowances should be made in the Design for rework without causing (or causing minimal) damage to the rest of the part or to other parts in the Product.

After the parts are produced, they will have to be tested, firstly, to see if they meet the Requirements of the Design and secondly, to see if they meet the

User Requirements. They will then have to be integrated into the next higher level assembly and so on until the top level assembly is complete. Therefore, it is necessary to have access points in the Design to prove that it works as expected and as required by the Customer. Additional circuits or fastening points may be required to achieve this. It is important that these be included in the Design right from the start so that there are no surprises causing extra work and re-design, thus increasing costs.

In addition to testing the Design to prove it works the way it is supposed to, testing or troubleshooting is required if it does not work. Failures can occur at any time during the Product's life. Therefore, access points to aid in locating the cause of the failure will help during the integration and test phase of the Product as well as after it is delivered to the customer. Sometimes the User Requirements specify a Mean Time to Repair (MTTR) in order to reduce downtime of the Product. If the test and troubleshooting features were added as an afterthought, the demands they would place on the Product's performance could limit the Design to the point that its User Requirements are no longer met. Therefore, the testability aspects of the Design should be included right from the start of the Project.

Coinciding with the testability of the Product are the Test Plans and Test Procedures. Before a Product is ready for delivery, there are two types of tests which usually have to be performed: Design Proving Tests or Qualification Tests and Acceptance Tests. Qualification Tests are used to show the Customer that the Design meets all the User Requirements. Some tests may or may not occur at lower levels in the Design depending on whether the tested characteristic is

visible or not at the top level. Some tests may be destructive in nature (e.g., some vibration or shock tests) and therefore may require a qualification model of the Product. Acceptance Tests are used by the Company and the Customer to prove that the Product is functioning correctly before delivery and therefore may be less extensive than the Qualification Tests.

Creating the Test Plan during the design phase may not only help in the testability of the Product but having it in place at some point before the Design is finished will help eliminate surprises during the test phase. There may be some physical constraints which prohibit having all test access points included in the Design. When choosing which points must be accessed and which should be (but may be left out), knowledge of the Test Plan during the design process will be invaluable.

Therefore, in order to keep costs down, it is necessary that the Designer pay attention to the following:

- producibility of the Design;
- where the part is going to be produced (in-house or not);
- efficiencies in production where the parts will be produced;
- rework with minimal damage;
- testability of the Design;
- troubleshooting;

and,

- test plans - qualification and acceptance

The MES should, therefore, help the Project Manager by referring to the various aspects above at the necessary times during the design process.

4.7 Documentation

Traditionally, the documentation aspects of the Design have been left to the end of the Project when everything has been more or less firmed up and is not likely to change. This may have been due to the technology of producing a document. It was manual and tedious and usually was very difficult to change after being laid out. Problems existed with that mode of operating. Designers left the Company or were assigned to a new Project and did not have the time to devote to the laborious process of documentation. Designers who were left to "clean up" did not understand some of the Product and created errors. Others simply forgot why or how something was done because it was some time since the subject was covered. But now, with newer technology such as word processors, CAD and automatic typesetters, the effects of a change are not so much of a problem and the approach should be different.

Due to the complexity of systems today, the importance of the documentation to both the Company and the Customer is high. During the Project, if a new Designer were to join the Design Team, the documentation that is currently available would quickly bring him/her up to date. The part of the Design that the Customer first sees before any hardware is around is usually some sort of documentation. The Marketing department would use any documentation that is available to prepare their brochures for the Product. Any future Customers would see some sort of documentation via the marketing

brochures first. After purchase and delivery of the Product, the Customer would look at the documentation frequently during setup, for training, when using it or when it requires maintenance.

When would one create the documentation necessary? The answer to this, would be, "all the time." Because of the electronic media available today, the Designers can start entering and saving data necessary for good documentation while creating the Design. After a Control Point, where a Design "freeze" takes place, the Design should be relatively stable and the data could be collated and prepared for when it is needed.

The logical portion of the design process provides mechanisms by which the Design can be documented while actively creating it (e.g., Context Diagrams, Data Dictionaries, Data Flow Graphs and Structure Diagrams). After the Design has been allocated to Hardware and Software, there is a different way in which the documentation is presented.

The hardware and software parts of the Design have similar types of documentation which are displayed in different ways. These types of documentation could include:

- system, functional and product specifications which aid in the design process by showing in a consolidated format the Design at various stages and at various levels of detail;
- Quality Assurance policies for manufacture and test;
- Quality Control and inspection procedures;

- test specifications, test plans, and test procedures which show how the Product is going to be tested;

and,

- operator's and maintenance manuals.

Due to their nature, there is also different ways in presenting hardware and software documentation. These have been separated for clarity.

4.7.1 Hardware Documentation

The hardware part of the documentation package includes many different types of drawings and should include descriptions of the operation of the different circuitry, block diagrams, family trees, timing diagrams, and any other diagrams or lists which help in describing how to produce, operate or maintain the Product. Drawings should include assembly drawings, parts lists, schematics, printed circuit board layouts and tracings, wire lists, wiring diagrams, specification control drawings, process control drawings and other drawings which aid in the manufacture and test of the Product.

4.7.2 Software Documentation

The software part of the documentation package could include software specifications, hierarchical drawings, state diagrams, flow charts, pseudo-code, source code listings, data lists, listings of test software used to test the software modules (if not the same as the delivered software), integration test software used when integrating the software with the hardware and any other information used to develop and test the software package.

4.7.3 Documentation Control

Documentation Control and Configuration Management are two similar aspects which help both the Company and the Customer in knowing what is being delivered. Documentation Control can be defined as the method by which one keeps control of the documentation used whereas Configuration Management can be defined as the method by which one controls the configuration of the Design (or the Product depending on whether it is being designed or being produced). When errors are discovered in a document which do not change the Design, it can be revised without changing the configuration of the Design (or Product). If for some reason there has to be deviation from the Design due to an error in manufacturing of some specific parts, the configuration of the Products which contain those parts would be different without affecting the Design documentation. Both are important in order that the documentation matches what is being produced.

4.8 Logistics Support

Traditionally, Logistics Support has been viewed as a function to be used only after the Product has been delivered and is in use by the Customer. As far as the Design was concerned, it was only added as an extra after the majority of the Design was complete.

Today, the insights that proper analysis of the whole Product's life cycle can give to the Designer is important. The reliability analyses and life cycle costs associated with the Product can have an influence in some areas of the Design. If, for example, in an online system, there were a certain part that was

key to the operation of the system and if it required replacing due to failure, the system would have to be turned off. In order for the system to have minimal downtime, there would have to be a replacement available on site (a spare part held only for replacement reasons). If this part were inexpensive, then one would not mind keeping a stock of these parts on hand. But, if it were expensive, and if it had a high probability of failure, one may wish to have either a different part used, a different Design incorporating a backup mechanism allowing replacement without turning the system off, or a different Design not having this part key to the operation of the system.

These tradeoffs (and their associated life cycle costs) are not possible if proper reliability analyses are not performed during the design process. The reliability affects not only the Customer's view of the Product but also of the Company as a whole. Therefore, it is important that the MES make reference to these at various stages throughout the design process.

4.9 Summary

In this chapter, several management issues were raised:

- the "Economics of the Design";
 - allocation of limited resources;
 - aspects of controlling the design Project;
- and,
- the role of the Customer in the Project.

The economics of the Design covered the costs of the design process and the effects they had on the price of the end product. The necessity of tracking costs was brought out and it was concluded that the MES should help the Project Manager address this.

The use of limited resources was covered and the impact it has on the costs of the Project as well as on the schedule that must be kept. It was pointed out that the parallel model helps in this respect because the resources can be interleaved to enable more use, lessen downtime and therefore lower the overall expense.

Control issues and the importance they had in keeping the costs down were discussed. A method of controlling a parallel design process was suggested where there was limited waiting time between phases (on the part of the Designers) but there was still management control of the Design. This involved the idea of Control Points or milestones and "Design freezes".

The importance the Customer has on the outcome of the Design was highlighted. Both the Customer and Management have an important part to play concerning the success of the design Project. If they are not keenly involved with its outcome, the Design may never get finished.

Various influences on the Design which are not directly part of the actual design process were also covered. These included:

- Quality Assurance;
- Production and Test;
- Documentation;

and,

- Logistics Support.

The Q.A. department not only ensures that a quality Product is delivered to the Customer but also can be a great help during the design process through their experience with specifications and the ways of conforming to them.

Production aspects of how and where the parts were to be produced were discussed. The related costs and ways of keeping them as low as possible were mentioned. The Test Plan and the Product's associated testability have an influence on the Design. Addressing this during the design process will help eliminate surprises when trying to prove the Design to the Customer and later deliver the Product.

The necessity of different types of documentation to both the Company and the Customer was highlighted. As well, the importance of the control of both the Product and its associated documentation were mentioned.

The idea of Logistics Support and its role in the design process was discussed. If one were to be able to properly predict the total life cycle effects of the Product, one could Design it for the maximum benefit of both the Customer and the Company.

All these issues, although they do not dictate how to design a Product, they have an influence on the Product and its associated costs (both monetary and reputation).

5.0 THE PARALLEL MODEL VERSUS THE ISSUES

In Chapter 3, the parallel model was first introduced. With it, came the ideas of modularity and modular consistency. These allowed the phases of a design process to be split in a parallel fashion without compromising the traceability of the Design back to the original User Requirements.

In addition, it was shown that each phase of the design process may be viewed as an operation having a set of antecedents, a set of consequences and taking a specified period of time. In this way, the design project may be mapped onto a Network Flow Graph and be handled by a set of Program Management Tools. From this, a critical path can be established on which a Project Manager can focus his/her efforts. In addition, resource loading and Project costs can be predicted and managed.

At the end of Chapter 3 as well as in Chapter 4, various issues which face a design Project were discussed. This Chapter will discuss the parallel model and how well it handles the issues raised in the previous chapters. Some of the issues, due to their nature, cannot be dealt with directly by the parallel model alone, but could be dealt with by a Project Manager while using the model as a guide. From this, the design of the Management Expert System can begin.

5.1 Process Flow Issues

These issues relate to the flow of the design process and how to keep it running as efficiently as possible. It was assumed in Chapter 3 that in order to have maximum parallelism, the design process must not be bogged down due to a single design event or a lack of a single resource.

Stoppages (and workarounds in general) can be treated in either of two ways: stop the process and correct the cause of the stoppage; or, adjust the process to accommodate the cause of the stoppage until it is corrected. Neither of these approaches suits every situation ideally, but the parallel model restricts the effect of the stoppage to only the modules directly affected. Therefore, this model would allow for a graceful degradation of the process flow, thus providing maximum efficiency.

The Project Manager would use the Management Expert System to try different scenarios before deciding which would best suit his/her particular needs and priorities. These scenarios would, in turn, only affect the modules already affected by the stoppage.

Changes in requirements or design, either generated by the customer or by the company can cause a disruption in the design flow. If there is a change in the requirements of the Design, the change would be examined to see what part of the design process (i.e., Context Diagram, Data Flow Graph, etc.) would have to be repeated or revised. Again, the parallel model would restrict the effect of the change to only the modules directly affected. Also, if the rest of the

Design were to proceed, the possible future effects to the design process would be minimized.

5.2 Management Issues

The management issues were divided into categories, each of which covered some part of the management aspect of the design process. In the following paragraphs the parallel model will be discussed with respect to how it helps the Project Manager deal with these issues.

Work in the parallel model is broken down in a modular fashion. Therefore, related costs can also be categorized in a similar work breakdown structure. A number of Program Management Tools on the market can relate the costs (both past and predicted) to this type of breakdown structure. Since the parallel model allows the design process to be mapped onto a Network Flow Graph, current Program Management Tools may be used to track costs related to the design Project. Serial methodologies, on the other hand, may not be as effective in this respect.

In Chapter 4, it was assumed that except for special circumstances, the Project Manager has limited resources. Consequently, he/she must make maximum use of the resources at hand and plan for the resources needed in the future. The parallel model allows operations to be staggered in order that a resource can be allocated to tasks not happening at the same time. This lessens the backlog of tasks being delayed while waiting for the use of the resource. Some Program Management Tools on the market allow the Project Manager to plan for the loading of the resources required. Therefore, the

parallel model has an advantage over the serial methodology in this regard also.

As was raised in the previous chapters, the Control Issues are an important part in the successful outcome of a design Project. As pointed out in Chapter 4, there is a time when all similar phases in the Design are complete. These points could be used as Control Points. Recommended Control Points would be at the end of the Context Diagrams, Loading Analyses, Structure Diagrams, Physical Designs, and the last test for meeting the User Requirements. Ideally, as in the serial methodology, one would like to have 100% of a phase complete before conducting a review at the Control Point. There are several reasons why this would not happen without having some part of the design process held up. Therefore, in the parallel model, the Project Manager would choose a level of completion for a certain phase (say 90% to 95%) before conducting a review.

During the review, a list of open items would be established including discrepancies in the Design and the tasks not fully completed. Any items not on the list of open items would be "frozen" and those parts would proceed "at risk" until all the opened items are dealt with. Attached to the list would be a schedule with completion dates. Following the last date on the list, a mini-review would be conducted in order to close off the open items, weigh the impact on the rest of the Design already proceeding, and "freeze" the Design up to that stage. This allows the Design to be reviewed without causing undue delay and frees valuable resources to carry on.

5.3 Risk Factor

As mentioned previously, for every project, there is a certain risk of the Design not being completed according to the User Requirements. The Risk Factor is defined as this risk in the form of a percentage. At the beginning of the Project, the risk is very close to 100% because, at this point, little is known about the Design. At the end of the Project, the risk should be about 2-3% since the Product is ready for delivery. Risk would only reach 0% when the Design has been in the field for a number of years and is a mature Product. Between the start and the end of the Project though, the risk would be somewhere between 100 % and 3%. Figure 4.2 illustrates both desirable and undesirable "routes" for the Risk Factor to take. In the following paragraphs, a proposed structure for determining what the Risk Factor is for the Project at any given time. A more definite structure should be determined only after the relationship between the Risk Factors and several actual Projects is studied.

To start with, there would be a base structure for the Risk Factor, upon which would be built the additional risk related to modules proceeding "at risk" and the number of items on the Open Item List. This base structure would be based upon the past milestone at which a part of the Design has been "frozen" and the next milestone where a Design Review must take place. The basic Risk Factors for the different milestones are as shown in Figure 5.1 and Table 5.1

Figure 5.1

Milestones versus Basic Risk Factors

Milestone	Risk Factor (%)	Approximate Completion (%)
Context Diagram	70	4
Data Flow Analysis	50	10
Structure Diagrams	35	20
Physical Design	17	48
Implementation	9	66
Testing	3	100

Table 5.1

Additional Risks would be as follows:

- for each open item on the Open Item List - add .5% - open items are treated as unknown parts of the Design and therefore increase the risk of the Project;

- for each module not completed in a design phase - add 1%;

- for each module not completed and left as an open item after a Design Review - add 2% - these open items cause greater risk to the Design if carried as open items;

and,

- for each module proceeding "at risk" before a Design Review - add 1% - there is a risk of re-design as a result of the Review, efforts would be better placed on modules not yet finished the phase.

As can be seen, there is a disincentive to proceed with either open items or modules proceeding beyond what is required for the next Design Review. On the other hand, there is an incentive to completing the Design up to a point of having a successful Review, thus "freezing" the Design to that point. For example, say the Data Flow Analysis Review were completed but Structure Diagram Review were not, the basic Risk Factor would be 35%. In addition, there were three modules not yet completed (+3%), one module not yet completed the previous phase (+2%), one module proceeding "at risk" (+1%), and a total of ten open items on the Open Item List (+10%). Then the Risk Factor for the Project would be 51%. If, in the same example, 50% of the funds were already used, looking at Figure 5.1, the Project Manager would see that the Project was starting to get in trouble since less than 20% of the Project is complete at a Risk Factor of still 51%.

5.4 Summary

In this Chapter, both process flow and management issues were summarized with the relationships they had with the parallel model. In this respect, the parallel model had the following advantages over a serial methodology:

- effects due to stoppages (and workarounds), for whatever the reason, would be minimized - the phases were divided into separate tasks according to the modular breakdown of the Design;
- because the parallel model is modular, the costs (both past and predicted) can be traced back according to the breakdown structure by a Program Management Tool;
- the limited resources can be more effectively allocated to different tasks (which are inherently staggered) in the parallel model;
- controls on the design process would not be lost - Design Reviews would take place at regular milestones and open items would be tracked in the form of Open Item Lists (which in the practical sense, are necessary anyway) - a method of calculating the Project Risk Factor was proposed to aid the Project Manager in determining the status of the Project;

and,

- the parallel model more closely emulates the actual work environment, can be tracked by standard Program Management Tools and therefore, would be more easily adhered to by those concerned.

6.0 THE SYSTEM

In previous chapters, the parallel design model was developed. In addition, various issues that present themselves at different times throughout the design process were discussed. Other issues were highlighted which are not directly related to the design process, itself, but have an impact on the Design during production. The Management Expert System should use the parallel model as a basis for guiding the Project Manager through the design process while addressing these issues (or at least highlighting them to the Project Manager) at the necessary points in the process.

In the normal top-down design philosophy, before one could start addressing the Kernel, one must first address the top level of the system. Since the scope of this thesis is limited to the Kernel, the Management Expert System will be described only to the extent that is necessary to design the Kernel.

Figure 6.1 shows the MES in a way that could be viewed by the User if he/she were able to see past the User Interface. The Technology Database, Program Management Tools (PMT) and Other System Tools would be available to the User for other purposes through the User Interface without directly using the MES (as shown in Figure 1.2).

The majority of the inputs to the MES would be through a dialog with the User. The User would choose a mode in which to enter and the MES would guide him/her through a series of questions and answers. Other data, such as a

completion of an outstanding item, may be input by addressing the items in an Open Item List. Besides data entry, the User may wish to query the system as to the status of the Project or propose a different scenario. This would result in new predictions, other comments or advice and other output data. If there were other on-line sources of information, there could be automatic "internal" updates of the Project. This is shown as a dotted arrow for future reference and will not be covered in this thesis for the sake of simplicity. Because the MES would be capable of handling concurrent activities in the work environment, the automatic updates could be added at a later date without too much trouble.

Figure 6.1

This Chapter will cover, in a little more depth, what could be used for the User Interface and the Program Management Tools if the System's design were to continue. Nevertheless, the descriptions in this chapter will not be as detailed as that of the Kernel because the rest of the design of the MES is beyond the scope of this thesis. The intent is to bring together the MES as a system so that the Kernel can be viewed in a working entity (even if the entity be only in the mind of the reader). Therefore, one must not limit the MES as only being capable of having the parts as described in this chapter, but should allow for the idea of future extensions that the Kernel is capable of operating within.

6.1 The User Interface

The User Interface is the User's link with the Management Expert System. It must be user friendly but still have enough power to efficiently control one's access to the MES. If the Project is going smoothly and only answering questions and Project status updates are necessary, relatively few entries would be required each day. Therefore, the MES should be efficient and easy to use so the Project Manager can make the updates without having to remember a lot of commands or how to use the System.

Therefore, for the purpose of this thesis, the User Interface would be an interface consisting of a series of windows along with a set of pull-down menus associated with each window and selection within a window. The opening window would require the User to select either a "new" or "current" Project which would cause the MES to enter the Initialization or Daily Startup mode, respectively. A choice of "new" would open a main window which would guide

the User through the set of questions for the mode and would result in a new Project Scenario. The pull-down menus in this mode would consist of:

a. - File/System access

- Open - used to open an existing file to base defaults on,
- Save - used to save current entries for later use,
- Print - used to print the current entries,
- Other System Tools - which the User may wish to access,

and,

- Quit - to stop the session;

b. - Initialization mode access

- Reset - reset to defaults,
- Design Complexity (questions) - used to answer questions or to change answers,
- Schedules (questions) - used to answer questions or to change answers,
- Cost or Risk (questions) - used to answer questions or to change answers,

and,

- Create Prospective Scenario (the method of displaying the Scenario will be described later),

c. - Change Scenario

- Open - used to open an existing file to base some part of a Prospective Scenario on,

- Close - close a file opened above,

- Save - save the Prospective Scenario for later use (this allows for multiple Scenarios to be tried before committing the results to the Project,

- Add - add a Task or set of Tasks to the Scenario,

- Cut - delete a selection for adding somewhere else in the Project,

- Delete - delete a selected Task from the Scenario,

- Update - update the Prospective Scenario Cost and Risk Factors and other data necessary for the Project,

- Change - change a Task name or other selected entry (e.g., slack remaining),

and,

- Continuous Running Mode - selection of this would put the MES through the Daily Startup mode and onto the Continuous Running mode automatically;

d. - Program Management Displays

- PERT chart,
- GANTT chart,
- cost breakout,
- milestone billings,

and,

- resource allocation;

and,

e. - Other System Tools

- Word Processor,
- Spreadsheet,
- CAD/CAE Tools,
- Modem,
- Program Management Tools,

and,

- Company's Local Area Network.

Figure 6.2

As the User answers questions, the MES would update the Particular database for the Project. When all the questions have been answered, the User would select "Create Prospective Scenario" which would command the MES to create a Prospective Scenario based on the answers given. If some answers were missed or there were no answers given, the default values would be used. The Prospective Scenario then would be displayed in the form of a PERT chart similar to that of Figure 3.3 in a format similar to that of Figure 6.2. This chart

would show each of the tasks broken out according to the hierarchical structure given by the User in the Design Complexity questions.

At this point, the User would modify the initial chart by adding task names, adding or deleting tasks where they are not necessary, etc. Once satisfied, the User would select "Update" to update the Cost and Risk factors associated with the Project and continue the modification process. If the User were basically satisfied with the Project Scenario, he/she would select "Continuous Running Mode" from the menu. This would command the MES to run through the Daily Startup mode, Create a Current Scenario based on the Prospective, and continue on to the Continuous Running mode.

For example, if in the case of the radar system, the User chose in the Design Complexity questions to have only two levels with three sub-modules in the second level (to start with), the preliminary Prospective Scenario would be:

- an overall Context Diagram Task for the system;
- Context Diagram, Data Flow Analysis, Structure Diagram, Physical Design and Testing Tasks for each of the three parts of the system (transmitter/receiver, processor and display);

and,

- a final Testing Task at the end.

Then, the User would name each of the Tasks according to the part of the system it was associated with. If he/she wished to further subdivide the transmitter/receiver into two parts but only after the Data Flow Analysis was

2

MICRO

complete (outgoing and incoming signals used the same waveguide and antenna), the User could do this by selecting the point where the division was to be made (Structure Diagram for transmitter/receiver) and select "Add Task". The following would then take place:

- a new Structure Diagram Task would appear - linked from the Data Flow Analysis Task for the transmitter/receiver but not linked back to the Network Flow Graph;

- the User would re-name the Tasks to "transmitter" and "receiver" and select a point to link the free end back to the Graph (if not chosen, the system would default to reconnecting at the Testing Task);

- he/she would then select "Update";

and,

- the MES would recreate the Prospective Scenario with the above revisions in place.

This process would continue until the User is satisfied with the prospective Scenario. Once satisfied, the User would select "Continuous Running Mode" from the menu and the MES would continue through the Daily Startup mode into the Continuous Running mode.

A choice of "current" during the opening window (Daily Startup mode) would cause the User Interface to request a Project file on which to base the Current Scenario. After a Project file is chosen by the User, the MES would

carry out the rest of the Daily Startup mode and automatically switch to the Continuous Running mode.

The main window for the Continuous Running mode would be the Current Scenario for the current Project. As can be seen in Figure 6.2, the Scenario is represented in the form of a Network Flow Graph. Various information important to the Project as a whole also would be displayed in the main window. This could include the critical dates, number of Open Items in the Open Item Lists, Risk Factor and Cost Factor. The critical path of the Project along with the Running Tasks on it would be highlighted. Also, the number of Open Items related to Tasks on the critical path would be stated. This is to alert the User in order that he/she may recognize and address these first. During the Question and Answer Session, the User would select a Task and "Open" it. This would produce a window containing the questions, Open Items, and answers along with other pertinent data for the particular Task. In order to address the Open Items in the main Open Item List or the Items on the critical path, the User would select them as he/she would select a Task.

Not shown in Figure 6.2, is the particular data for the individual Running Tasks that would also be included in the main window. As shown in Figure 6.3a, this data would include the following:

- the Task's name;
- the Task's number - this associates the Running Task with the other Tasks in the Project;
- the type of Task;

Labelling of Tasks

Figure 6.3

- the number of Open Items assigned to the Task;
- the Latest finish time - this is the latest time the task can be finished before schedule slippage occurs;
- estimated time to complete;

- slack time

- if the Task is not started - the default would be equal to a percentage of the time allowed to complete the Task based on the information given in the Initialization mode (this could be shortened (but not lengthened) during the Prospective Scenarios routine if the User wished),

- if the Task is started - it is equal to the total time left for the Task minus the estimated time required to complete the Task;

and,

- latest start time - equal to the latest finish time minus the estimated time to complete and the slack.

If a task were selected by the User, it would be blackened as shown in Figure 6.3b and if it were finished, it would be shaded as in Figure 6.3c. Also, if the Task were started or finished, the start date and finish date would be displayed instead of the estimated dates.

The window for Prospective Scenarios would be similar to that of the Current Scenario except that the User would be allowed to change certain data or add and delete Tasks. There also would be a window to allow the User to change the Generic or Particular databases.

Pull down menus for the Continuous Running mode would include:

a. - File/System access

- Save - used to save the Current Scenario for later use (entries are usually saved but it would be safer to consciously save the Current Scenario periodically),

- Print - used to print the Current Scenario or the selected window,

- Other System Tools - which the User may wish to access,

and,

- Quit - to stop the session;

b. - Change Design Database

- Change Generic,

and,

- Change Particular;

c. - Questions and Answers

- Open - open selected item,

- Close - close selected window,

- Change Answer - change selected answer,

- Split - split task,

and,

- Continue - finish selected Task and carry Open Items to next Task;

d. - Prospective Scenarios

- Open - used to open an existing file to base some part of a Prospective Scenario on,

- Close - close a file opened above,

- Save - save the Prospective Scenario for later use (this allows for multiple Scenarios to be tried before committing the results to the Project,

- Revise - create Prospective Scenario based on the Current,

- Add - add a Task to the Scenario,

- Cut - delete a selection for adding somewhere else in the Project,

- Delete - delete a selected Task from the Scenario,

- Update - update the Prospective Scenario Cost and Risk Factors and other data necessary for the Project,

- Change - change a Task name or other selected entry (e.g., slack remaining),

- Make Current - create new Current Scenario based on the Prospective,

and,

- Make Current Anyway - create new Current Scenario based on the Prospective (even though a "Changed Current Scenario" flag indicates that the Current Scenario changed since the Prospective was first created);

e. - Program Management Displays

- PERT chart,

- GANTT chart,

- cost breakout,

- milestone billings,

and,

- resource allocation;

and,

f. - Other System Tools

- Word Processor,

- Spreadsheet,
 - CAD/CAE Tools,
 - Modem,
 - Program Management Tools,
- and,
- Company's Local Area Network.

As shown in the pull-down menus, the MES would provide access to Other System Tools as desired by the User. The selection of one of these Tools would, in turn, create another window within which the User would access and use the Tool. The number of Tools capable of being accessed and the capabilities therein would be dependent upon the host computer and the Tools, themselves. The User Interface would just provide access to them without having to exit the MES.

6.2 Program Management Tools

The Program Management Tools for the Management Expert System would help the MES by taking the Project Scenario data and transforming it into a form that would be useful for a Project Manager. As stated in Chapter 1, the PMT's would consist of some sort of linear optimization program to generate data to be presented in the form of PERT or GANTT charts. These Tools could be accessed either directly by the User or indirectly through the Kernel.

The output of the PMT's should be in a form in which the Kernel can glean information about the critical path and other parts of the Project which would be built upon to give the Project Manager the whole Scenario (Current or Prospective). Since the PMT's have not been developed in this thesis or a set of existing PMT's have not been chosen yet, the exact interface between the Kernel and the PMT's cannot be discussed.

6.3 Summary

In this Chapter, Management Expert System's User Interface and Program Management Tools were discussed. These were presented to enable the reader to understand more fully how the MES would operate and what kind of environment might be suitable for the Kernel.

It was suggested that the User Interface consist of a set of windows each with pull-down menus to allow the User easy access to the MES without having to remember a lot of commands. The Scenarios would be presented in the form of Network Flow Graphs while highlighting the critical path of the Project. Other System Tools also could be accessed without having to leave the MES environment. The Program Management Tools were briefly discussed to present what would be required of them in order for the MES to function.

7.0 THE KERNEL

In the last Chapter, there was a brief description of how the User would see the MES along with some mention of how it would be used. This Chapter will focus more deeply at the Kernel which is at the heart of the System. It controls how the system operates and is the means by which the User takes advantage of the system's functions.

Based on the issues presented and the points made in the previous chapters, a Set of Requirements for the Kernel (as would be perceived by the user) will be defined. For the purpose of this thesis, actual processing, data flow efficiency, and other concrete requirements will not be discussed because the structure and control of the design process is more important than the efficiency of the MES. The Design of the Kernel will be developed from this Set of Requirements. Then, the Kernel's operation will be presented with Procedural Flow and Structure Diagrams. Finally, the Kernel will be reviewed showing how its Design meets its Requirements.

For additional reading, pseudo-code can be found in Appendix A and typical questions to be asked in the different design phases (part of the Generic Database) can be found in Appendix B. (Note: The tasks shown in both the Structure Diagrams and the pseudo-code are for the purpose of the reader to perceive how the Kernel models the parallelism in the work environment.

Therefore, many of the tasks shown could be better represented as sequential procedures when actually committing the Kernel to source code.)

7.1 Requirements Definition

Before the design of the Kernel can take place, one must understand the Requirements. A subset of the Management Expert System's Requirements would be directly applicable to the Kernel. Others would have to be known to create the proper context for the Kernel. Since the issues (both Design and Management) were addressed in previous chapters, they will be summarized here with reference to where they were addressed (by page and/or by paragraph number). References will be in brackets and will look like (para 3.1) or (page 45). In order that the MES meets a Project Manager's needs, the Kernel has the following requirements.

7.1.1 Design Process Requirements

It shall:

- a. - accurately follow the parallel model, which in the case of this thesis is the parallel version of the methodology developed by Bowen and Brown (para 3.1);
- b. - guide the Project Manager through the above design process and present him/her with the necessary information to manage the design Project;
- c. - create an environment for entering data and retrieving information;

d. - help the Project Manager reduce the effects of stoppages (para 3.2.1) and therefore increase the overall flow of the design process;

e. - show the Project Manager the effects of additions or reductions of resources (para 3.2.3);

f. - help the Project Manager in tracking open issues in the Design and alert him/her when they will start affecting the critical path of the Project (para 3.2.5);

and,

g. - maintain a changeable database of questions to be asked at various stages in the design process - this makes the MES flexible enough to be modified if not suitable for a particular person or project.

It should:

a. - show the Project Manager the parts of the Design that are affected by a change in its requirements (paras 3.2.3 and 3.2.4);

and,

b. - suggest alternatives in the process when correcting errors in the Design or adjusting to changes in requirements (paras 3.2.3 and 3.2.4).

It may:

a. - allow for automatic inputs of raw data from other sources besides the User.

7.1.2 Management Requirements

It shall:

a. - help the Project Manager in making best use of his/her resources by showing cost/benefits of different scenarios (para 4.2);

b. - help the Project Manager in controlling the design project by establishing milestones where Design "freezes" can take place (pages 48-49 and 70) thereby reducing the risk of the overall Project;

c. - maintain a list of open items and discrepancies found during Design reviews, each with associated risks and costs (pages 49 and 70);

d. - maintain a list of tasks (each with their associated risks) who are proceeding "at risk" while there still remains items in c. above (page 49);

and,

e. - maintain a total risk factor of the Design during the course of the Project (pages 49-51).

It should:

a. - help the Project Manager in predicting the associated costs of the Project and of the end product (para 4.1);

b. - build a database of past costs in aid of a. above (para 4.1);

and,

c. - track outstanding requests and other approvals requiring a response by the Customer (para 4.4).

7.1.3 Other Requirements

It shall:

a. - interface with the rest of the Management Expert System (Program Management and other system tools);

b. - alert the Project Manager of the necessary producibility/testability aspects at various times throughout the Project (para 5.2);

c. - alert the Project Manager to consult the Q.A. department at the necessary parts of the design project (para 5.1);

and,

d. - make reference to various logistical aspects of the Product at the necessary stages in the design process (para 5.4).

7.2 The Kernel Design

The Kernel would be comprised of four basic modules (Figure 7.1):

- the Design Model which would guide the User through the design process;
 - User Interface Buffer which would be the "port" through which the User Interface would receive data from the Kernel;
 - Data Input Buffer which, in the future, would allow automatic methods of entering data into the Design Database or Current Scenario;
- and,

- the Program Management Interface which allows the Design Model to gain access to the Program Management Tools resident in the MES.

Kernel's Breakdown Structure

Figure 7.1

7.2.1 Design Model

The Design Model has its own database as well as four basic modules. These are:

- the Allocator/Scheduler task;
- the Running Tasks (e.g., Context Diagram, Data Flow Analysis, etc.);
- the Scenario package

and,

- the Database Control task.

The Allocator/Scheduler task, according to the Current Scenario, sets up and controls which are active during any particular part of the design process. Before gaining access to any Running Task, the User Interface must gain access to the Allocator/Scheduler task in order to identify which of the tasks are active and ready for use. This task is active during all of the modes of the Design Package (Initialization, Daily Startup and Continuous Running).

The Running Tasks are probably the mainstay for the MES (Note: These Tasks are for illustrative purposes only and show how the Kernel models the parallelism in the work environment. These should be converted to procedures when committing the MES to source code.) Each of these Tasks have assigned to them:

- a Particular part of the Design (e.g., in the case of the radar system - the display);

and,

- a Generic phase of the design process (e.g., Data Flow Analysis).

They would guide the User through the phase of the design process for the Particular part of the Design assigned to them. In keeping with the parallel model, even though there is a specific number of generic Running Tasks, during any particular time in the design Project, any number of these Running Tasks in any configuration could be guiding the User through their assigned part of the Design. For example, two Physical Design tasks could be functioning within one area of the Design while a Data Flow Analysis and three Structure Diagram tasks could be working with the User in other areas. In this way, the Kernel can properly model the actual work environment and guide the Project Manager with relevant information.

During normal operation, these Tasks would draw the necessary information from the User by a series of questions and answers, update their part of the database as to which questions have been answered, update the Current Scenario as to their current status, and based on the answers, continue with the design process as deemed necessary. Unanswered questions, or questions left to a later date, would be highlighted in their particular database for future use. These would have to be completed in a satisfactory manner before moving on to the next phase in the process. Some could be carried, though, only if a major milestone were not crossed and if the answer had no implications outside of the module. These outstanding items would result in an increase in risk due to the added chance for re-design. Any successfully

answered questions result in a decrease in risk and a continuation of the process.

The Scenario package has two basic types of Scenarios: Current and Prospective. Each of these has a stored database in the event of a power failure or that the session with the User is finished (in the case of an offline type of single user system). In addition, it has Cost and Risk Factor procedures which associate factors with different phases of the design process, levels of complexity within each phase and percentage completion of each phase. These factors would be able to be modified by the User to allow him/her to tailor the system to the work environment.

The Current Scenario routine keeps track of the process flow and the progress made within each Running Task on the Project. With this information, it can estimate the percentage completion of each task, flag delays or potential risk areas to the User and indicate the items which are on the critical path in order that the more critical items may be concentrated on. To do this, it uses the Risk Factor routine, which calculates the potential risk left in completing the Project, and the PERT chart set up by the Program Management Tools. The Current Scenario routine also keeps track of the open items and assigns them to the proper tasks in the design Project. It would also show the Project Manager the cost to complete the Project based on the Current Scenario.

The Prospective Scenario routine helps the Project Manager by performing cost/ benefit tradeoffs on different scenarios proposed by the Project Manager. At the beginning of a Project, this routine would help the Project Manager in setting up the "new" Scenario while during the Project, it would

Revise a copy of the Current Scenario. This routine could implement the changes necessary by using the Cost and Risk Factor routines along with the Program Management Interface. Once satisfied, the User could command the MES to convert the Prospective Scenario desired into a new Current Scenario.

The Design Database consists of two parts: the Generic and the Particular. The Generic database consists of a part related to the Project in general and a part containing the different sets related to the phases of the design process. Each generic set consists of the antecedents required in order to start the Running Task without undue risk, the questions asked during the particular task and a set of consequences of the successful completion the task. If the Project Manager wished to proceed without a given antecedent, there would be an added risk factor associated with it. The questions asked during a particular task are set up in a dependence/precedence relationship (linked list or tree structure) in order to guide the Project Manager through the particular set. Similarly, the sets are linked to enable the Kernel to model and follow the design process flow.

The Particular part of the Design Database keeps a running status of which questions have been asked, which have been answered satisfactorily, and which have been left open to be answered at another time. With this part, the Running Tasks can keep track of where they are and how much of their part of the design process is complete.

The Database Control task controls the access to the Design Database and keeps it organized. Each Running Task, when first activated, asks the Database Control task to set a Particular area for it to keep its status (of

questions asked, etc.) and asks for access to its Generic set of data. This is to keep its status separate from other Running Tasks of the same generic type while maintaining the Generic information in one location. In the event that the User wishes to make changes to the Generic part of the database, they are also made through the Database Control task.

7.2.2 User Interface Buffer

The User Interface Buffer is a buffer through which parts of the Kernel can send data to the User Interface. In this way, several parts of the Kernel can send data and not have to wait for a response. Data would be represented by the source of the data (e.g., Running Task), destination (where in the User Interface the data is applicable), and the data itself.

7.2.3 Program Management Interface

The Program Management Interface is a buffer through which the Prospective Scenario routine and the Program Management Tools can send and receive data. This interface is used because of the nature of PMT's to allow the Kernel and the PMT's to operate independently without have to wait for the other to respond.

7.3 The Kernel Operation

The MES can be in one of three modes of operation:

- Initialization - when the particulars of a Design can be entered as a starting position for the design project;

- Daily Startup - when the system is started up at the beginning of the day or the start of a session;

or,

- Continuous Running - in continuous use during the daily operation of the MES.

In both the Initialization and Continuous Running modes, the User has an interactive session with the system where data is entered and retrieved as well as a background operation where the Design Model runs and interacts with other areas of the system (i.e., the Program Management Tools).

When the MES is first turned on, the User would be asked if the Project were a "new" or a "current" one (an old file is just a "current" Project which has been completed). A choice of "new" would start the MES into the Initialization mode. Upon termination by the User of the Initialization mode or by a choice of "current", the MES would activate the Daily Startup mode. Upon completion of the Daily Startup mode, the MES would automatically switch into the Continuous Running mode.

Figure 7.2 shows the Kernel's structure. Calls to the Kernel from other parts of the MES would be through the Design procedure (or in the future, also through the Data Input procedure) and would typically be from the User Interface. Based on the mode of operation and on the information received, the Design procedure would decide which procedure of the Design Model should be called. The User Interface Buffer and the Program Management Interface are buffers to allow the Design Model to operate without having to wait for the other parts of the MES to respond. The Program Management Tools, for example, could take a relatively long time to complete their tasks. The User Interface could be busy with the User, possibly using another system tool.

Kernel Structure

Figure 7.2

No interfaces or buffers are evident for the Technology Database or for the Other System Tools. The Database would be a set of organized files on disk or other storage media and the Tools accessed by the Kernel would be

editors, disk drivers and other modules which are part of the operating system. Since the MES only uses these indirectly (after compilation) and the User could use them (or gain access to them) outside of the operation of the MES, interfaces are implied in Figure 6.1 but are not shown in Figure 7.2.

At any time, but more particularly during the Continuous Running mode (because it would be running the majority of the time), the User may wish to end the session. In this case, the Kernel would command the modules to stop in the following order:

- Data Input task;
 - Design Model package, by calling the Continuous Running procedure;
 - Program Management Interface;
- and,
- User Interface Buffer.

7.3.1 Initialization

This is the mode whereby the Project Manager enters data at the start of a Project in order to initialize the Particular databases for the "new" Project. Upon the User gaining access to this mode, the MES would, through the User Interface and the Initialization procedure, activate the Allocator/Scheduler task, the Database Control task and the Prospective Scenario package. These

modules would, in turn, start their initialization routines making themselves ready to respond to necessary calls or rendezvous.

After the Allocator/Scheduler task is initialized, it would control the creation of a Particular database for the "new" Project. Figure 7.3 shows how data would flow in order to achieve this.

Figure 7.3

The Allocator/Scheduler task would get the first set of questions from the Generic Database (via the Database Control task) and send them to the User

via the User Interface Buffer. Replies from the User, in the form of answers to questions, would then come back through the User Interface and the Initialization procedure to the Allocator/Scheduler task. The Particular database for the Project would be updated and more questions would be sent to the User. Once all the questions were answered, the User would have the opportunity to change his/her answers or create a Prospective Scenario. The Procedural Flow for creating a Prospective Scenario will be discussed in paragraph 7.3.4. The resulting Prospective Scenario or the results of actions would, then, be sent back to the User for review. After reviewing the results, the User could change his/her answers, add or delete Tasks and re-create the Prospective Scenario. Once satisfied, the User would proceed to the Daily Startup mode.

Figure 7.4

Figure 7.4 shows the structure of the Design Model as it would be in the Initialization mode. Upon a call to the Initialization procedure during the startup of the MES, the Database Control and Allocator/Scheduler tasks would be activated and the Scenario package would be started (①). Then, the Allocator/Scheduler task would make a series of calls or rendezvous (②, ③ and ④) while creating the desired Prospective Scenario and the Particular database for the "new" Project. Finally, when all is complete, the Particular database is made ready for Daily use, the Initialization mode is terminated and the Daily Startup mode is entered.

The following are some typical inputs (part of the Generic database) that are required from the User during the Initialization mode (Note: All default values are arbitrary and only set as a starting point. Provision is made in the Continuous Running mode to change the default values according to a company's particular operating practices.):

1. - to give the MES a rough idea of the complexity of the Design -
 - a. - set defaults to actual data of a "current" (or older) Project?
(defaults based on actual data of a "current" (or older) Project of similar complexity could give the MES a truer picture of the complexity of the "new" Project);
 - b. - number of hierarchical layers in the Design (default - 4);
 - c. - number of sub-modules per module in each layer
(default - 4);

d. - maximum number of lines of source code per lowest level module (default - 100 lines);

and,

e. - approximate number of hardware components per lowest level module;

2. - for estimation of schedules -

a. - set defaults to actual data of a "current" (or old) Project?

b. - time required to fully develop source code (default - 1 manhour per line);

c. - time required to fully develop hardware (default - .5 manhours per component);

d. - types of resources needed to develop modules;

e. - number of each type of resource used including people (default - 1);

f. - estimated start date;

g. - delivery deadline;

h. - percentage slack in schedule for slippage (default - 10% of time per design phase - minimum 1 week);

and,

i. - percentage completion of design phase before Design Review and milestone (range 80-100%, default 95%);

and,

3. - for estimation of costs or risks-

a. - set defaults to actual data of a "current" (or old) Project?

b. - cost of material per lowest level module (default - \$100);

c. - cost of resources per type (default - \$0);

d. - labour costs.(default - \$50 per hour);

e. - risk per question left as an open item (default - .5%);

and,

f. - maximum risk allowed to be carried by an individual module to the next phase of the design process in the form of open items (default - 3%).

7.3.2 Daily Startup

In this mode, the MES starts the necessary tasks and packages of the "current" Project for the daily use by the Project Manager (or other User). Upon the User gaining access to this mode, the MES would first activate (if not already activated by the Initialization mode) the Allocator/Scheduler and the

Database Control tasks and the Scenario package. These modules would, in turn, start their initialization routines making themselves ready to respond to necessary calls or rendezvous.

Figure 7.5

After the Allocator/Scheduler task is initialized, it would call the Current Scenario (Figure 7.5) to determine which Running Tasks should be currently operating and, therefore, should be activated. In the case of the radar system example, say the Data flow Analysis was still being performed on the processor but the rest of the system was at the Structure Diagram phase. Then the Data Flow Analysis Task for the processor and the Structure Diagram Tasks for the the transmitter, the receiver (note they are separate now), and the display would

be activated. The Running Tasks, once activated, would rendezvous with the Database Control task to gain access to their Particular database which contains a list of unanswered questions. Any questions, during a previous session, that the User had left for a later date would be flagged. In addition, any unanswered questions which were past their due date, would have a further notification. Any Running Task being activated for the first time, would request the Database Control task to allocate space for its Particular database and a list of generic questions which would have to be answered. All the Running Tasks would receive, from the Generic database, the next set of actual questions to be asked.

Figure 7.6 shows the structure of the Design Model as it would be in the Daily Startup mode. Upon a call to the Daily Startup procedure, the Database Control and Allocator/Scheduler tasks would be activated and the Scenario package would be started (①). After the Allocator/Scheduler task is initialized, it would get the current project status from the Current Scenario (②) and then activate the necessary Running Tasks (③). Once activated, the Running Tasks would get their necessary data from the Database Control task (④). At this point, the Daily Startup mode would be terminated and the Continuous Running mode would be entered.

Design Model (Daily Startup)

Figure 7.6

7.3.3 Continuous Running

For the majority of the time that the MES is in use, it would be in the Continuous Running mode. This mode has three sub-modes of operation:

1. - changes to the Design Database (either Generic or Particular) for both the Initialization or the Continuous Running mode - since the majority of time would be spent in the Continuous Running mode changes could be made at this point for either mode;

2. - answering questions and making updates to the project status - assuming everything proceeds as planned, the majority of time would be spent in this sub-mode;

and,

3. - creating Prospective Scenarios for possible incorporation into the "current" Project - in case there is a change of plans or if a redesign is required.

In addition, since this mode is in use for the majority of time, it contains the routine for stopping the Design Model at the end of a session. The Continuous Running procedure would stop the modules contained in the Design Model in the following order:

- Allocator/Scheduler task;
- all the Running Tasks;
- Scenario package;

and,

- Database Control task.

7.3.3.1 Changes to the Design Database

In this sub-mode, the User, through the User Interface can gain access to the Design Database in order to make changes. Changes to the Generic database would be those that would be consistent for every project within the company. These could be changes to the default values for the Initialization mode (e.g., labour costs) or deletion/addition of questions for the Initialization

mode or for a particular phase of the design process. Changes to the Particular database would be those necessary for the particular Project (e.g., changing an answer to a question previously answered incorrectly).

Figure 7.7

Figure 7.7 shows how the flow would be in this sub-mode. Only the changes to the Generic database will be discussed since changes to the Particular database would be similar in nature. The User Interface, through the Continuous Running procedure, would gain access to the Database Control task and would ask for a copy of the item to be changed. The Database Control task would, in turn, retrieve it from the Generic database and send it to the Continuous Running procedure. The item would then be sent to the User Interface Buffer for review by the User. Once it's changed, the item along with reference to where in the Generic database it should be placed, would be sent back by the User Interface, through the Continuous Running procedure, to the Database Control task.

7.3.3.2 Answering Questions and Project Status Updates

In this sub-mode, the dialog between the User and the Running Tasks takes place. Upon entering this sub-mode, the User starts the process. but the Running Task in which he/she is in dialog with would be in control of the flow. The other tasks and modules in the Design Model would operate in aid of the Running Task.

Figure 7.8 shows the flow as it would be in this sub-mode. The User would choose which Running Task he/she wished to work with. The User would know which were available because, in the Daily Startup mode, the Current Scenario was sent to the User for review. Then, the User, through the User Interface, would answer an outstanding question, send an update of the status of a part of the Project or ask for further questions. The User Interface, through the Continuous Running procedure, would gain access to the Allocator/Scheduler task to find the Running Task desired. The Allocator/Scheduler task would, in turn, get the current status of the Project from the Current Scenario routine. Then, it would return the location of the desired Running Task. Next, the answer, status or request for questions would be sent from the Continuous Running procedure to the Running Task concerned.

The Running Task would do any of three things:

1. - send the answer to the Database Control task for updating the Particular database;

2. - send the status to the Current Scenario for updating - the Current Scenario would, in turn, make the update and send it to the User Interface Buffer for review by the User;

Procedural Flow
(Questions or Status)

Figure 7.8

or,

3. - send a request for further questions along with its Running Task type and assigned designation (to distinguish it from other similar Running Tasks) to the Database Control task.

Sometimes, an answer would also require an update to the status. In which case the Running Task would carry out both 1. and 2. above. In the case of 3. above, the Database Control task, would get the status from the Particular database concerned, along with the Generic questions and send the next set of questions (and status of questions remaining) back to the Running Task concerned. The Running Task would, in turn, send the next set of questions (and status) to the User Interface Buffer for review by the User.

If a Running Task had its last question answered, and it were finished, it would automatically call the Current Scenario, give its final status and terminate itself. On the other hand, the User may wish to leave some questions unanswered and keep them (with an associated risk) as open items in the next task. If this were deemed possible by the Running Task, it would send the associated risk to the User. If the User still wished to terminate the Running task, it would call the Current Scenario, give its final status and terminate itself. Periodically, the Allocator/Scheduler task would call the Current Scenario to check for Running Tasks that need to be activated, and would, in turn, activate them.

Other operations that the Running Tasks carry out during this sub-mode can be grouped together and classified as Process Flow Logic. These include:

- determining if the antecedents of the Running Task have been met before proceeding with the rest of the Task;

- logic for dividing Tasks into smaller tasks as the work on the project gets to a lower level in the Design;

and,

- determining if the consequences of the Running Task are sufficient to proceed to the next phase of the design process;

The way of determining the necessary antecedents and consequences would be buried in the Generic set of questions for the type of Running Task involved. This has the advantage of allowing the Project Manager to be able to determine which are necessary and which are not (but are nice to have). Also, associated risks can be applied to the questions if they are not properly answered.

The logic for splitting tasks would be contained in the Running Task, itself. Since methods for combining the work at the end of the Project would be contained in the questions for the generic Running Task "Testing", it would not be necessary to include these in the logic contained in the Running Task, itself. The rules for splitting a Running Task would be:

1. - the work must be split along a boundary between two distinct sub-modules of the Design;

2. - the necessary antecedents would be met for each of the Sub-Tasks assigned to the sub-modules of the Design;

and,

3. - any interfaces between the sub-modules of the Design are well defined.

Once the new Sub-Tasks are defined, the original Running Task would update the Current Scenario showing the new Running Tasks operating in place of the original. Then, the original Running Task would terminate itself.

Appendix B has some typical questions and Project status requirements to be followed for each of the phases of the design process.

7.3.3.3 Prospective Scenarios

The Prospective Scenario sub-mode provides the User with a means by which he/she can change the Current Scenario during the course of a Project. There can be many reasons for wishing make a change (e.g., behind schedule, redesign, change of requirements, etc.). If during the testing phase of the display (in the radar system example) the display did not perform as expected, then an analysis would be performed to determine the cause of the error. In this case, the Project Manager may wish to try several Prospective Scenarios to determine which route to take in order to solve the problem. This could mean re-designing a part of the display while continuing testing the rest of the system "at risk", continuing with the error until the next generation or revision of the

Product is developed or relaxing the requirement so the Product reflects what is expected.

Figure 7.9

In this sub-mode, the User, through the User Interface task and the Continuous Running procedure, would gain access to the Prospective Scenario routine. As shown in Figure 7.9, upon a command to create a new Prospective Scenario, the Continuous Running procedure would call the Prospective Scenario routine to create one. The first Prospective Scenario would be a copy of the latest Current Scenario and a date/time tag to indicate when the Prospective Scenario was first created. Then, a series of changes would take

place, each showing the overall Cost and Risk Factors associated with the changed Scenario.

When the User is satisfied with the Prospective Scenario, a command would be sent through the User Interface task and Continuous Running procedure to the Current Scenario routine, to change the Prospective Scenario to Current. The Current Scenario would, in turn, check the date/time tag to see if the Current Scenario has changed since first entering this sub-mode. If it has changed, a flag stating this, along with the latest Current Scenario would be sent to the User Interface Buffer. Then, the User would decide to change the Current Scenario anyway or to continue changing the Prospective Scenario as necessary. Once the User is satisfied, the command to change the Current Scenario is sent again. The Current Scenario, then, gets the scenario data from the Prospective Scenario, updates itself, and sends the new Current Scenario to the User Interface Buffer for review by the User.

7.3.3.4 Continuous Running Structure

As shown in Figure 7.10, the structure of the Continuous Running mode follows the three sub-modes. In the first sub-mode, the Continuous Running routine would gain access to the Database Control task in order for the User to be able to change the Generic database (①a).

In the second sub-mode, the Continuous Running procedure would gain access to the Allocator/Scheduler task which in turn would get the Current Scenario and send back the location of the desired Running Task (①b and ②b). Then, the Continuous Running procedure would gain access to the

desired Running Task (③) in order to answer a question, update the status, or request more questions. The Running Task, in turn, would either send the answer to the Database Control task for updating the Particular database (④), send the status to the Current Scenario for updating (⑥), or get the next set of questions from the database (through the Database Control task) and send them to the User interface Buffer (④ and ⑤). Any changes to the Current Scenario would result in the new one being sent to the User Interface Buffer (⑤) for review by the User.

Design Model (Continuous Running)

Figure 7.10

carry out the rest of the Daily Startup mode and automatically switch to the Continuous Running mode.

The main window for the Continuous Running mode would be the Current Scenario for the current Project. As can be seen in Figure 6.2, the Scenario is represented in the form of a Network Flow Graph. Various information important to the Project as a whole also would be displayed in the main window. This could include the critical dates, number of Open Items in the Open Item Lists, Risk Factor and Cost Factor. The critical path of the Project along with the Running Tasks on it would be highlighted. Also, the number of Open Items related to Tasks on the critical path would be stated. This is to alert the User in order that he/she may recognize and address these first. During the Question and Answer Session, the User would select a Task and "Open" it. This would produce a window containing the questions, Open Items, and answers along with other pertinent data for the particular Task. In order to address the Open Items in the main Open Item List or the Items on the critical path, the User would select them as he/she would select a Task.

Not shown in Figure 6.2, is the particular data for the individual Running Tasks that would also be included in the main window. As shown in Figure 6.3a, this data would include the following:

- the Task's name;
- the Task's number - this associates the Running Task with the other Tasks in the Project;
- the type of Task;

and,

- the Scenario Database Control task.

The Prospective and Current procedures would not get called since the User Interface, through the Continuous Running procedure, would be active in the shutdown of the MES. As well, the Data Input Buffer and the Running Tasks, which also call the Current procedure, would have been shut down prior to the Stop procedure being called. The Cost and Risk Factor routines would not have to be independently shut down since they are passive and the Prospective Task and the Current procedure are the only modules which call them.

7.3.4.1 Prospective Scenario

The Prospective Scenario routine interacts with the Program Management Tools. Since these Tools take a relatively long time to perform their duties (especially on a large Project), the Prospective Scenario routine requires an internal task to carry out the routines and be capable of waiting for the Program Management Tools to finish. Meanwhile, the rest of the MES (and the User) can carry on with other tasks. Inside the Scenario package, the Prospective Task would interface with the Cost and Risk Factor procedures and the Scenario Database Control task. Outside the Scenario package, the Prospective Task would interface with the Program Management Interface, the User Interface Buffer and the Design Model's Database Control task.

This Scenario has two basic routines:

- create a New Prospective Scenario (Initialization mode of the MES),

and,

- develop Prospective Scenario based on the Current Scenario (Daily Startup and Continuous Running modes of the MES).

Creating a New Prospective Scenario involves:

- getting the particular information for the Project;
- setting up the initial Project data in the Prospective Scenario Database,
- formatting the Scenario information and getting the PMT's to organize the data in a Project flow chart (e.g., PERT chart);
- applying Cost and Risk Factors to the Project;

and,

- sending the information to the User Interface Buffer for review by the User.

Figure 7.11 shows the flow during the creation of a new Prospective Scenario. When the Prospective procedure is called to create a new Prospective Scenario, it would transfer the necessary information to the Prospective Task, which in turn, would control the creation of the New Project Scenario. The Prospective Task would get the particular Project data from the Particular database and the generic information about the design process through the Design Model's Database Control task.

Figure 7.11

The Prospective Task would organize the data and ensure the Project followed the general process flow governed by the parallel model. It could do this because:

- contained in the particular Project data would be the hierarchical structure and other information about the Design (e.g., number of

hierarchical layers, number of sub-modules per module in each layer, resources assigned to each model, etc.)

and,

- contained in the generic information would be the necessary links between the generic phases of the design process.

In addition, the Prospective Task would organize the other information about each module in a format suitable for the Program Management Tools, save it in the Prospective Scenario Database via the Scenario Database Control task, and send it to the PMT's (via the Program Management Interface) for the creation of the first Project flow chart. After the Project Scenario is created, it would be sent back through the Program Management Interface to the Prospective Task.

Cost and Risk Factors, based on the particular data, would be applied to the Project Scenario by the Cost and Risk Factor procedures and the modified Project Scenario would be stored in the Prospective Scenario Database through the Scenario Database Control task. Then, the new Prospective Scenario would be sent by the Prospective Task to the User Interface Buffer for review. The User would then have the opportunity to modify the information to the particular needs of the Project. This would happen in a similar way to that of developing a Prospective Scenario based on the Current.

Developing a Prospective Scenario based on the Current Scenario involves:

- getting the particular information for the Project if it has changed since the last Prospective Scenario;

- copying the latest Current Scenario into the Prospective Scenario Database along with a date/time tag for future reference and sending it to the User for review - this is meant to get the process started since the User ultimately wishes to have a "changed" Current Scenario without affecting the process of updating the status of the Project;

- getting the Revised Scenario from the User;

- using the Program Management Tools to re-organize the data;

- applying Cost and Risk Factors to the Revised Project Scenario,

and,

- storing the Revised Prospective Scenario and sending it to the User for review.

Figure 7.12 shows the flow during this process. To start this process, a command to create a Prospective Scenario based on the latest Current Scenario would come through the Prospective procedure to the Prospective Task. The Prospective Task would, in turn, get any new particular data from the Particular database which would have been changed since the creation of the

last Prospective Scenario. This would be done by gaining access to the Design Model's Database Control task.

Procedural Flow
(Prospective From Current)

Figure 7.12

The Prospective Task would send to the Scenario Database Control task, the new Project data along with a command to make a Prospective Scenario based on the latest Current Scenario. Next, the Scenario Database Control task would copy the latest Scenario from the Current Scenario Database to the Prospective Scenario Database and apply a date/time tag of the time it was copied for future reference. Then, the Prospective Scenario would be sent back to the Prospective Task in order to be forwarded to the User Interface Buffer for review by the User (the MES's Continuous Running mode) or the Program Management Interface for updating (the MES's Daily Startup mode).

After the Prospective Scenario has been changed by the User, the changed Scenario would be sent back through the Prospective procedure to the Prospective Task where it would be checked for conformance to the parallel model and forwarded to the Program Management Interface. The PMT's would re-organize the data into a Project flow chart. After the Revised Project Scenario is created, it would be sent back through the Program Management Interface to the Prospective Task. Cost and Risk Factors, based on the particular data, would be applied to the Project Scenario by the Cost and Risk Factor procedures and the Revised Project Scenario would be stored in the Prospective Scenario Database through the Scenario Database Control task. Then the new Prospective Scenario would be sent by the Prospective Task to the User Interface Buffer for review by the User.

Figure 7.13 shows the structure of the Scenario package as it would be while creating Prospective Scenarios.

Figure 7.13

In the case of creating a new Prospective Scenario, the Prospective procedure would send the necessary data to the Prospective Task (①). The Prospective task would get the particular Project data from the Design Model's Database Control task and store it in the Prospective Scenario Database via the Scenario Database Control task (②, ③ and ④). Then, the data would be organized into a Project flow chart by the Program Management Tools (⑤), Cost and Risk Factors applied and stored in the Prospective Scenario Database (⑥ and ④). The Prospective Task would get the new Prospective Scenario and forward it to the User for review (③, ④ and ⑦).

In the case of creating a Prospective Scenario based on a Current Scenario, the Prospective procedure would send the necessary data to the Prospective Task (①). The Prospective task would get any of the changed

particular Project data from the Design Model's Database Control task (②). Then, it would command the Scenario Database Control task to send a copy of the latest Current Scenario to the Prospective Scenario Database (③,⑧ and ④). If this were first time through in the MES's Continuous Running mode, the Prospective Scenario would be sent to the user for review (⑦). If this were in the Daily Startup mode, the new Prospective Scenario would be sent to the PMT's for updating (⑤).

On the next time through, the changed Scenario would be sent through the Prospective procedure to the Prospective Task and forwarded to the PMT's for updating (① and ⑤). After the Scenario data came back from the PMT's, Cost and Risk Factors would be applied and the new Prospective Scenario would be stored in the Prospective Scenario Database (③,⑥ and ④). Then, the Prospective Task would get the Prospective Scenario in order for it to be sent to the User for review (③,④ and ⑦).

7.3.4.2 Current Scenario

The Current Scenario routine, unlike the Prospective Scenario, is a means by which the MES keeps the status of the Project and updates it at the necessary times. Since it does not have to wait a relatively long time for any other procedures or tasks to carry out their duties, there would be no need for a separate task to carry out the bulk of the work. Inside the Scenario package, the Current procedure would interface with the Cost and Risk Factor procedures and the Scenario Database Control task. Outside the Scenario package, the Current procedure would interface with the calling procedure or task and the User Interface Buffer.

The Current Scenario has three basic routines:

- create a new Current Scenario based on the Prospective Scenario (Daily Startup and Continuous Running modes of the MES);

- update the status of the current Project (Continuous Running mode of the MES);

and,

- get the current status of the current Project (Continuous Running mode of the MES).

Figure 7.14 shows the flow for the routines of the Current Scenario. Upon a command to create a Current Scenario based on the Prospective Scenario, the Current procedure checks the data/time tag of the Prospective Scenario. This is to determine if the Current Scenario had changed since the Prospective Scenario was first started. If there had been a change, the Current procedure would get the latest Current Scenario (via the Scenario Database Control task), attach a message stating that the Scenario had changed and send it to the User Interface Buffer for response by the User.

If the Current Scenario had not changed, the MES were in the Daily startup mode or a command to disregard the data/time tag had come, the Current procedure would command the Scenario Database Control task to copy the Prospective Scenario data into the Current Scenario Database. Then, a copy of the Current Scenario would be sent to the User Interface Buffer for review by the User.

be entered into the Current Scenario Database (via the Scenario Database Control task). Again, the new Current Scenario would be sent to the User Interface Buffer for review by the User.

Upon a command to get the status of the Current Scenario, the Current procedure would get the latest Current Scenario via the Scenario Database Control task and, depending on the command, either return it to the calling task or forward it to the User Interface Buffer.

Figure 7.15

Figure 7.15 shows the structure of the Scenario package as it would be while creating and updating the Current Scenario. In the case of creating a Current Scenario from a Prospective, the Current procedure would command the Scenario Database Control task to copy the necessary data from the Prospective Scenario Database to the Current Scenario Database (①, ② and ③). The Current Scenario would then be sent to the User Interface Buffer (④).

In the case of updating the status of the Current Scenario, the Current procedure would forward the data to the Cost and Risk Factor procedures and then on to the Scenario Database Control task to be input into the Current Scenario Database (①, ⑤ and ③). Upon termination of a Running Task, the status along with the open items to be forwarded to the next Task would be sent to the Scenario Database Control task to be input into the Current Scenario Database (① and ③). Then, the Current procedure would get the new Current Scenario (via the Scenario Database Control task) and send it to the User Interface Buffer for review by the User (①, ③ and ④).

In the case of getting the current status, the Current procedure would get the new Current Scenario via the Scenario Database Control task (① and ③) and either send it back to the calling task or to the User Interface Buffer for review by the User (④).

7.3.4.3 Cost and Risk Factors

The Cost Factor procedure allows the MES to determine the cost of the Project. The Total Cost of the Project would be equal to the actual costs plus the predicted costs. The actual costs are the costs which have been incurred during the execution of the different tasks of the Project. The projected costs are those which will be encountered before the end of the Project. These would be calculated based on the Particular data initially set out in the Initialization mode or as changed in the Continuous Running mode of the MES (e.g., manhours per module times the costs per manhour, costs of material, etc.).

The Risk Factor procedure allows the MES to determine the Total Risk left in the design Project. This factor would be in the form of a percentage risk and would decrease with the increasing maturity of the Design. Ways of decreasing the Total Risk would include:

- crossing a milestone in the Project and successfully finishing a Design review;

- closing out open items left as a result of a Project Manager wishing to proceed "at risk";

and,

- not letting one part of the Design get too far ahead of the rest.

Ways of increasing the Total Risk would include:

- allowing parts of the Design to proceed past a milestone where a Design review would take place (there could be a possibility of re-design);

- continuing on to another phase of the Design with open items (open items could have a major impact on the outcome of the Design);

and,

- continuing on with the Design without having a Design Review (a Design review tends to tie the Design together and uncover issues which may otherwise have been forgotten.

The Risk Factor procedure would determine where the milestones are in relation to the progress of the Project. These would be typically when a design phase is complete (or mostly complete). This would allow the MES to determine whether a Design Review is due and then would alert the Project Manager, if necessary. The Risk Factor procedure would use the desired Scenario for which the risk is to be taken and do the following:

1. - determine where the Running Tasks are in relation to the milestone;

2. - determine if a Design Review had taken place or not;

3. - determine if there are open items as a result of the Design Review;

and,

4. - calculate the Total Risk based on the above.

7.4 Validation

The requirements for the Kernel were presented in paragraph 7.1. This section is organized in a similar fashion. In the following paragraphs, each of the requirements will be examined and an argument will be presented showing the reasons why the Kernel meets its requirement.

7.4.1 Design Process Requirements

"It shall accurately follow the parallel model, which in the case of this thesis is the parallel version of the methodology developed by Bowen and Brown."

The Prospective task in the Scenario package of the Kernel models the design flow during the Initialization mode of the MES. As well, the Prospective task checks any Scenario modified by the User and organizes it in accordance with the design process flow before sending it to the Program Management Tools. The nature of different Running Tasks guiding parallel operations is consistent with the required parallel model. For the case of a single-user system, the User can only respond to one menu at a time, and for the case of a multi-user system, the calls could be queued. Therefore, it is not necessary for the actual tasks in the Kernel to be running concurrently, just for it to be perceived by the User to be running that way. The Running Tasks shown in Structure Diagrams and the ADA based pseudo-code imply concurrency, but are meant only to be illustrative of how the User would perceive the System.

A changeable Generic database allows, if desired, another suitable type of design methodology to be used instead of that by Bowen and Brown. Due to the generic nature of the Running Tasks and the way in which the Kernel handles them, a completely new set of design phases could be substituted for those given in Appendix B. This gives added flexibility to the MES.

(paras 7.3.1, 7.3.3.1 and 7.3.4.1)

"It shall guide the Project Manager through the above design process and present him/her with the necessary information to manage the design Project."

Depending on the style of each Project Manager, the necessary information for one may not be sufficient for another. Therefore, incorporated in the Kernel's Design Database is a dependence/precedence list of questions to be answered during both the Initialization and Continuous Running modes of the MES. The set of questions in the Continuous Running mode is divided into the different phases of the design process. Each set guides the User through the particular phase of the Project in which it appears. All questions can be added to (if the User wishes to insert a question, or set of questions), modified, or deleted (if the User feels the question is not suitable or unnecessary). Therefore, the MES can be tailored to suit the particular Project Manager's needs. (pages 99 - 100 and 102, and para 7.3.3.1)

"It shall create an environment for entering data and retrieving information."

The question and answer sessions provide this environment. In addition, other updates to the status of the Project also may be entered during the Continuous Running mode. (paras 7.3.1 and 7.3.3.2)

"It shall help the Project Manager reduce the effects of stoppages and therefore increase the overall flow of the design process."

The parallel nature of the Design Model allows the Project to proceed, albeit less efficiently and sometimes "at risk", if one part of the Design is lagging

behind or is stopped. The Prospective Scenarios allow the Project Manager to try alternative ways of completing the Project and see the risks and costs involved with each alternative. (para 7.3.3.3 and pages 130 - 134)

"It shall show the Project Manager the effects of additions or reductions of resources."

The Prospective Scenarios offer the Project Manager a method of determining which best suits his/her needs in a cost effective manner.

(paras 7.3.1, 7.3.3.3, and 7.3.4.1)

"It shall help the Project Manager in tracking open issues in the Design and alert him/her when they will start affecting the critical path of the Project."

Open Item Lists attached to each Running Task as well as an overall Open Item List for the Project offer the User a method of tracking the outstanding issues in the Project. The Current Scenario procedure would determine which are the critical items and would alert the User by flagging them.

(pages 101 and 119)

"It shall maintain a changeable database of questions to be asked at various stages in the design process. This makes the MES flexible enough to be modified if not suitable for a particular person or project."

The Design Database is the changeable database of questions to be asked at various stages in the design process. The User would change questions and other Generic or Particular data during the Continuous Running mode of the MES. (pages 102 - 103 and para 7.3.1)

"It should show the Project Manager the parts of the Design that are affected by a change in its requirements."

This system does not show all the parts of the Design which are affected by a change in its requirements because of the fact that it does not link the Design directly. Linkages between modules of the Design exist indirectly, to the extent that work on them is connected by a network flow graph (PERT chart) in the form of tasks. Any delays in one task (due to redesign, etc.) may affect the critical path of the Project, affect the priorities of the Project Manager, and therefore, affect another task which may originally have been on the critical path. Also, because of the interdependence of some Designs and the nature of the process flow, changes to a completed task may affect the antecedents of other tasks and therefore, the status of the overall Project. Although direct links between modules of the Design are not in the MES at present, they could be implemented if the design of the MES were to continue. The Prospective Scenario routine allows the Project Manager to adjust the Project as necessary and the MES will show him/her the consequences of the actions taken.

(paras 7.3.3.3 and 7.3.4)

"It should suggest alternatives in the process when correcting errors in the Design or adjusting to changes in requirements."

This system does not suggest alternatives in the process although it would be nice to have. The design of MES does not preclude one from adding this feature in the future if desired and the processing power is available.

"It may allow for automatic inputs of raw data from other sources besides the User."

The MES allows for automatic inputs, if desired in the future, by having Data Input tasks and procedures. (pages 98 and 105)

7.4.2 Management Requirements

"It shall help the Project Manager in making best use of his/her resources by showing cost/benefits of different scenarios."

The Prospective Scenarios offer the Project Manager a method of determining which best suits his/her needs in a cost effective manner. (paras 7.3.1, 7.3.3.3, and 7.3.4.1)

"It shall help the Project Manager in controlling the design project by establishing milestones where Design "freezes" can take place thereby reducing the risk of the overall Project."

This is controlled by the Risk Factor procedure. If a certain percentage of a design phase is complete (percentage determined by the User during the Initialization mode), the Risk Factor procedure would alert him/her that a Design Review is necessary. (page 140)

"It shall maintain a list of open items and discrepancies found during Design reviews, each with associated risks and costs."

An overall Open Item List for the Project offers the User a method of tracking the outstanding issues in the Project. The User would enter the open items onto the list and the Current Scenario procedure would keep track of

them. The Risk Factor procedure determine the risk assigned to the Project as a result of these open items. (pages 139 - 140)

"It shall maintain a list of tasks (each with their associated risks) who are proceeding "at risk" while there still remains items in the Open Item list above."

The Risk Factor procedure would carry this out. (pages 139 - 140)

"It shall maintain a total risk factor of the Design during the course of the Project.

The Risk Factor procedure would carry this out.
(pages 101 and 139 - 140)

"It should help the Project Manager in predicting the associated costs of the Project and of the end Product."

The Cost Factor procedure would carry out the predicting of the costs of the Project. Since the MES does not directly deal with the Design itself, costs of the end Product would not be predicted. At various stages in the Question and Answer sessions, though, the MES would ask the Project Manager to perform this task. If in the future, the MES were linked to the Design itself by a structured design tool of some sort, the MES may be capable of offering this feature.
(pages 101 and 138)

"It should build a database of past costs in aid of the above."

The MES does not build an independent database of past costs but past costs would be recorded for reference by the Project Manager in files of past Projects. The Design of the Kernel does not preclude an implementation of this if desired in the future.

"It should track outstanding requests and other approvals requiring a response by the Customer."

This would be covered in the Open Item Lists as unanswered questions.

7.4.3 Other Requirements

"It shall interface with the rest of the Management Expert System (Program Management and Other System Tools)."

The Kernel does this through the use of Design and Data Input procedures and Data Input task for inputs and User Interface Buffer and Program Management Interface for outputs.

(page 98 and 103, and paras 7.2.2 and 7.2.3)

"It shall alert the Project Manager of the necessary producibility/ testability aspects at various times throughout the Project."

This would be carried out in the Question and Answer sessions of the Continuous Running mode. (para 7.3.3.2)

"It shall alert the Project Manager to consult the Q.A. department at the necessary parts of the design Project."

This would be carried out in the Question and Answer sessions of the Continuous Running mode. (para 7.3.3.2)

"It shall make reference to various logistical aspects of the Product at the necessary stages in the design process."

This would be carried out in the Question and Answer sessions of the Continuous Running mode. (para 7.3.3.2)

7.5 Summary

In this Chapter, the Design of the Kernel was developed. To begin the Chapter, the Set of Requirements for the Kernel was defined. This included design process and management as well as other requirements for the Kernel. Along with each requirement, there is a reference to where in the thesis the requirement was drawn from.

Based on the Set of Requirements, the Design was presented showing the hierarchical structure of the Kernel along with the general functionality of each routine or module. Then, there was a demonstration of how the Kernel would operate. Procedural Flow and Structure Diagrams were used to illustrate the Kernel's operation. This included the three main modes of operation:

- Initialization - where the initial setup for the Project took place;
- Daily Startup - where the daily power-up routine took place;

and,

- Continuous Running - where the majority of the operation of the MES took place.

The Continuous Running mode consisted of the three sub-modes:

- changes to the Design Database - which enabled the User to tailor the MES or the Project to his/her needs;

- answering questions and Project status updates - the routine process and tracking of the Project;

and,

- creating Prospective Scenarios - which enabled the User to change the Current Scenario as the Project required.

Then, the operation of the Scenario package was discussed. This included both the Prospective and the Current Scenario routines and how the Cost and Risk Factors were applied.

Once the Kernel was designed, arguments were presented showing how the Design of the Kernel met its Set of Requirements. This was done by quoting a requirement followed by the reasons why the Kernel did or did not meet it. References were included to show where in the Chapter the requirement was developed.

8.0 SUMMARY

This thesis has taken an existing methodology and expanded it in a parallel fashion to more closely model the work environment. This parallel model was built upon to create a design of a Kernel for a Management Expert System. To start with, the methodology, as presented by Bowen and Brown, was briefly discussed in Chapter 2. It was of a serial or pipeline nature, and although it covered the subject of Structured Design in a reasonable manner, it was felt that a parallel version would exhibit improved efficiency before its implementation into the Kernel of the MES.

In Chapter 3, the parallel model was developed. This model established parallelism consistent with the modular nature of a properly structured Design. What came out of this was a model which was more efficient because faster tasks of the design process could proceed independently of the slower ones. This would provide a more efficient allocation of limited resources and allow for re-design without the rest of the Design necessarily being held up.

This, at first, seemed to be a great improvement over the serial model but, now, greater controls would have to be implemented in order that the Design may be controlled to completion. A number of other issues were raised, all of which have some impact on the outcome of a Design and its development into a Product.

These issues included:

- the end of Chapter 3 (design process flow issues)
 - stoppages,
 - resources,
 - changes in requirements,
- and,
 - false technical assumptions;

and,

- Chapter 4 (management and other issues)
 - economics of the Design,
 - allocation of resources,
 - control issues,
 - the role of the Customer,
 - quality assurance issues,
 - producibility and testability issues,
 - documentation issues,
- and,
 - logistics support.

The parallel model and the issues raised were brought together in Chapter 5 to compare the model against the issues it would have to address in the design environment. Also in Chapter 5, a proposed method of determining the Risk Factor of a Project was discussed.

Chapter 6 went on to discuss the User Interface and the Program Management Tools. This enabled the reader to better understand the environment in which the Kernel could operate. The topics covered in this Chapter were suggestions in order to show the Kernel in a working entity. Therefore, one should not limit the MES only to the topics covered.

Chapter 7 took the parallel model along with the various issues and focused more deeply on the Kernel which was to be used as the basis for the Management Expert System. First, a Set of Requirements for the Kernel was proposed. Then, the Kernel was developed based on this Set of Requirements. Finally, arguments were presented showing how the design of the Kernel met its requirements.

In Section 8.1, the parallel model will be reviewed for its feasibility in a design environment. In Section 8.2, the design of the Kernel (and the MES, where suitable) will be reviewed for its potential for use on a design Project. In Section 8.3, conclusions will be made as a result of the work carried out to date. Possible extensions and further development of the MES will be discussed in Section 8.4.

8.1 Review of the Parallel Model

Even though the parallel model may be more difficult to control because of its increased complexity, it has the following advantages:

- effects due to stoppages, for whatever the reason, would be minimized because of the modular consistency with the Design;

- because the parallel model is modular, the costs (both past and predicted) can be traced according to the breakdown structure of the Project by a Program Management Tool;

- the limited resources can be more effectively allocated to different tasks which can be staggered in the parallel model;

- open items would be tracked in the form of Open Item Lists;

- a Project Risk Factor helps the Project Manager in determining the status of the Project;

and,

- the parallel model more closely emulates the actual work environment, can be tracked by standard Program Management Tools and therefore, would be more easily adhered to by those concerned.

Using this thesis as an example, one could not proceed as far as the Structure Diagrams for the Kernel (much less, some of the pseudo-code) if one had to strictly follow a serial methodology. In this way, a particular aspect of the MES (namely the Kernel) could be focused on to demonstrate a direction and the feasibility of the overall Design. Of course, some effort had to be made in areas of the Design other than the Kernel. Nevertheless, the only effort required was that which was necessary to create a proper context for the Kernel. Because the design of the Kernel has proceeded so far ahead of the rest of the System, there is risk of re-design. But if efforts had to be spread amongst all

aspects of the MES equally and if the Design were not feasible, then more time would be wasted than was absolutely necessary.

In any case, if the parallel model were automated in the form of a Management Expert System, the Project Manager would have increased control (rather than less control) on a more powerful methodology. Therefore, the parallel model is not only feasible but is necessary to properly emulate the actual design environment.

8.2 Review of the Design

In this section, the design of the Kernel (and the MES, where applicable) will be discussed. To begin with, various general issues will be covered. Then, a discussion of the items missing from the Design will take place. These will be accompanied with the reasons why they have been left out of this thesis and the risks involved in doing this.

8.2.1 Performance

The pseudo-code and the Structure Diagrams were based on a liberal interpretation of ADA. Because of this, one could easily get caught in the trap of discussing the performance of ADA, itself, and not the performance of the Kernel or the MES. ADA was chosen for two reasons: its ability to illustrate concurrency, and its suitability for the translation of Structure Diagrams directly into pseudo-code. To this end, the performance of ADA will not be covered.

Whenever the performance of a Design is discussed, there usually is concern about the issue of deadlock and whether or not it exists in the Design.

There are times when the MES is waiting for a response from the User or the Kernel is waiting for the Program Management Tools to finish. These may result in the appearance of a "hang-up" in the System. The majority of functions carried out by the MES are as a result of a command or other direction from the User and are carried out until completion. Potential places for deadlock could be:

- between the Kernel and the User Interface because of the feedback of information from the Kernel to the User while the User Interface is busy making another call to the Kernel;

or,

- between the Kernel and the PMT's because of the interchange of Scenario information between them.

Both of these have been freed by the use of buffers containing timeouts to limit the period of time in a given "wait state". Therefore, in the case of the MES, deadlock does not exist.

The requirement for the MES to react in "real time" does not present a problem because usually entries would be far enough apart to allow the System to react. If the User has to wait a long time between making entries into any system, he/she could get frustrated and would tend not to use it. This waiting period is dependent upon the patience of the User which tends to become shorter with increased usage of the particular system. Since the PMT's tend to take a relatively long time compared to the rest of the MES, there is a buffer to free the Kernel for other commands or directions by the User. Also, the User,

through the proposed User Interface, could gain access to other management tools while he/she is active within the MES. Therefore, if the PMT's took a long time or if there were a long wait period between entries into the System, the User need not leave the MES environment.

The design of the Kernel does not preclude its use in a multi-user environment, or in the future, automatic entries made by some other part of the network (e.g., completion of a task using CAD/CAE tools). The use of ADA for the pseudo-code implies that:

- procedures and passive packages are re-entrant and, therefore, can be used by different "callers" at essentially the same time;

and,

- tasks can operate independently and concurrently - because of the times involved, it would not be necessary to have true concurrency but to simulate it by time slicing or other multi-tasking scheme.

The overhead involved in the implementation of the MES is dependent upon the complexity of the MES desired. The overhead of the MES would be much greater in a multi-user system with two or more Users than if the MES were a single-user system with a foreground task (i.e., the User Interface) and many background tasks and procedures. Also, if it were a single-user system, much of the concurrency would not be necessary. Therefore, many of the tasks in the Kernel could be converted into procedures (e.g., the Running Tasks), thus reducing the overhead involved with separate tasks. Nevertheless, the

increased overhead involved in a multi-user system would be in the User Interface and is beyond the scope of this thesis.

8.2.2 System Integrity

The integrity of the MES can be looked at from two different perspectives:

- as it exists as presented;
- and,
- as it would be if the design of the MES were complete.

As the Design exists at present, there remains some items to be determined (Parts missing from the Design will be discussed in paragraph 8.2.3). For the most part, a structured approach to the Design was adhered to. For the purpose of this thesis, though, the efforts focused quickly on the design of the Kernel. This caused some structured design rules to be broken which normally would not have been. Therefore, since large areas of the Design (e.g., User Interface, Program Management Tools, etc.) have been left for later, there remains a risk of having to re-design some parts of the Kernel. For example, since the Design and Scenario Databases require more work before they are complete, the design of their respective controllers is not clear and also requires some work. In any case, the fact that there is a controller is important to this thesis because it involves the structure of the Kernel's design.

The choice of the type of system (i.e., multi-user, single-user, automatic input, etc.) has been left for later. This choice was not necessary until later because the User Interface was not being designed in detail nor were the total User Requirements specified. The design of the Kernel is flexible enough to be

tailored to whichever type of system is chosen. Because of this, the integrity of the Kernel is sound and of the MES as a whole could be quite sound (depending on the outcome of the rest of the Design).

8.2.3 Items Missing From the Design

Since this thesis is centered about the Design of the Kernel, a number of items have been left out. Other items, which are part of the Kernel, were made simple or were left for the future. This was all in the interest of conciseness and clarity (They may have confused the discussion of the structure and functionality if they were included). The following is a list of parts of the Design which have been left for later along with the reasons why and the risks involved by not including them.

The User Interface - Many User Interfaces are in use today and there are probably just as many opinions about them. Because of this, a whole thesis could be centered on this one issue. Therefore, in order that the Design of the Kernel be focused on, a User Interface was described briefly but was left for future development. Once the full set of User Requirements are specified, the design of the User Interface may begin. This may involve some changes to the User Interface buffer, but because there is a buffer, changes to the rest of the Kernel would be limited.

The Program Management Tools - Linear optimization tools of different sorts are available on the market today. Again, the choice of one of these would depend on the particular needs and desires of the person continuing with the Design and, therefore was left for future development. Once these Tools are

chosen, a more extensive design of the Scenario Database is required (which may involve some re-design of the Kernel).

The Scenario Database Structure - The structure of the Scenario Database and the types of data contained therein is highly dependent upon the PMT's chosen for the MES. Therefore, this has been left for when the PMT's are chosen.

Cost and Risk Factor details - Greater detail is needed in order that these Factors become truly reflective of the actual design environment. The design of a proper model could be a subject of a thesis by itself. In this thesis, the need for these factors was acknowledged but an arbitrary base with linear perturbations was used for the sake of simplicity. A more definitive base with a mix of linear and non-linear perturbations is required.

A Data Flow Analysis - A Procedural Flow Diagram was used instead of a Data Flow Graph for the purpose of illustrating the operating modes of the MES. This should be added to when the rest of the System is designed to create a Data Flow Graph. With this, a proper Data Flow Analysis may be performed. A full loading analysis and partitioning exercise was not carried out. It was felt that the major load on the processor would be by the Program Management Tools (not part of the thesis) and that the Program Management Interface would relieve the MES of a major holdup. Because this analysis was not performed, there is risk of re-design.

- A Data Dictionary was intentionally left out of the Design because, for the purpose of the thesis, it was felt that a proper structure in the

Design was more important and more illustrative of the functionality of the Kernel. A Data Dictionary may have confused the issue because some parts of the MES were not yet defined (e.g., User Interface, Program Management Tools). Once proper Data Flow Graphs are in place, the development of the Data Dictionary could take place. It is acknowledged that a Loading Analysis and Data Dictionary are necessary in order for the design of the Kernel (and MES) to continue.

A breakout of the Physical Design process - The Physical Design process was not broken out in a similar way to that of the Logical Design. It was felt that a breakout of just the Logical Design process was sufficient to illustrate a point and that a breakout beyond that of Physical Design and Testing was not important to the design of the Kernel. Further breakout can be carried out later since this would only require new generic types of Running Tasks. Consequently, a number of issues raised in Chapters 3 and 4 would have to be incorporated at the time when the Physical Design and Testing phases are further broken out.

The MES "learning from experience" - The subject of a system being capable of learning from experience was beyond the scope of this thesis. However, the System can be changed if the User finds it necessary to add another step in the design process or to change questions in the Generic database.

Building a database of past costs and risks for reference in the future - Costs and Risks are saved from past Projects and from Scenarios of current Projects. But, aside from editing them (a User Interface function) from one

Project to the next during the Project Scenarios sub-mode, there is no clear way of carrying this out. Again, it is left up to the User to learn and to take the past into consideration. This feature would be useful in a system which "learns from experience".

Suggesting alternatives to the design process when changing the Project Scenario - This feature has been left for the future and would probably coincide with the "learning from experience" features mentioned above.

Links in the Design of the Product to show interdependence of the modules - This feature was not added to the MES because the requirements were primarily to develop a management system and not a design system. Clearly, the design of a Product cannot be divorced from the management of the Project, but this feature was deemed to be beyond the scope of this thesis. If this System were made as an overlay or as an extra feature of a CAD/CAE system, this feature would be handy to have.

8.3 Conclusions

In order for any system to be useful for the Project Manager, there must be a Design model based on a structured methodology. This structure not only gives direction for the Project, but also allows the Project Manager to follow and track the flow of the design process. The Kernel of the MES is based on a structured model which is a parallel version of the methodology developed by Bowen and Brown.

Controls must be placed on the design process to allow the Project to be tracked by the Project Manager. A parallel methodology has the tendency to take away the control points or milestones that exist naturally in a serial or pipeline version. Therefore, care must be taken to ensure that these points remain and outstanding items are tracked. The parallel model keeps these milestones (between phases) in place with a formal review associated with each. Any outstanding items are tracked on various Open Item Lists. In addition, Open Items which are on the critical path are highlighted in order that they can be addressed first. Automation, in the form of the MES, takes the work involved with the new complexity away from the Project Manager in order to allow him/her to focus efforts elsewhere. Nevertheless, the control of the Project still rests with the Project Manager whereas the MES is just a tool for him/her to use. Good control (on the part of the Project Manager) is indicated by a consistently decreasing Risk Factor throughout the Project. On the other hand, poor control is indicated by a widely fluctuating Risk Factor together with increased costs and schedule slippages.

The Kernel is versatile and, therefore, can be used in various types of systems. Because of the ability of the Kernel to handle essentially parallel operations, it can be:

- based in a multi-user type of environment;
- used as an overlay or add-on feature to a CAD/CAE system for the control of a design Project;

- used as a single-user application in a multi-user environment with automatic inputs from other sources;

or it can be,

- simplified to be used in a single-user environment on a desktop personal computer.

The process flow is governed by the questions and directions held in the Generic part of the Design Database. This Database is changeable which allows the MES to be tailored to meet the particular needs and desires of the Company. If desired, another suitable type of design methodology may be used instead of the one chosen. In any case, Cost and Risk Factors along with Open Item Lists would still apply.

Because of the above, the MES has flexibility, can change with time, and the Project Manager can benefit from the experience gained in past projects. Therefore, the Management Expert System is a viable solution for the control of a design Project.

8.4 Extensions

In order to complete the MES and make it into a working system, there remains the following tasks:

- the definition of the rest of the User Requirements concerning the other parts of the MES (i.e., Single-user/Multi-user environment, etc.);

- further development of the User Interface;

- the choice of PMT's and the development of the interfaces to them;
 - the design of a more accurate Cost and Risk analysis;
 - a proper Data Flow Analysis including a proper loading analysis and a Data Dictionary;
 - further development of the Design database (including questions);
 - the choice of a target computer and the environment within which the MES has to operate;
 - the programming and debugging of the MES;
- and,
- the preparation of documentation and the creation of a Product.

GLOSSARY

Allocator/Scheduler task - A task in the Design Model which sets up and controls which Running Tasks are active in the package.

CAD - Computer Automated Design

CAE - Computer Automated Engineering

Cost Factor routine - A routine of the Scenario package which determines the projected costs for the Project.

Critical Path - A path in a PERT network which requires the longest time to complete

CSSR - Cost/Schedule Status Report

Current Scenario - The scenario which shows the actual sequence of events of the design Project.

Current Scenario routine - A routine in the Scenario package which sets up and controls the Current Scenario.

Database Control task - A task in the Design Model which controls entries into and retrievals from the Design Database.

Deadlock - The condition of a concurrent computer system where two or more active tasks are waiting for each other to respond causing inactivity amongst the tasks concerned.

Design Database - A database in the Design Model which contains both Generic and Particular data for the design project. It is organized by operating mode of the MES and by phase of the design Project.

Design Model package - A package in the Kernel which models the design process of the work environment.

GANTT - A graphic representation of data in the form of bars.

Kernel - The heart of the Management Expert System which controls the transformation and flow of data necessary to perform the System's functions.

Linear Programming - The field of applied mathematics where a linear function is maximized (or minimized) subject to a set of linear constraints

MES - Management Expert System

MTTR - Mean Time to Repair

Network Flow Graph - A linear programming structure having a source and a sink whereby one can use linear optimization techniques to obtain maximum flow rates and minimum execution times

Open Item List - A list of open items in the design Project which have yet to be addressed.

PERT - Program Evaluation and Review Technique

PMT - Program Management Tools - Tools which organize project data into a Network Flow Graph.

Program Management Interface - A task of the Kernel which buffers information between the Kernel and the Program Management Tools.

Prospective Scenario - A prospective version of the Current Scenario used to try a Scenario before implementation.

Prospective Scenario routine - A routine in the Scenario package which sets up and controls the Prospective Scenario.

Re-entrant - The characteristic of a routine whereby it can call itself or can be called by another caller before completion.

Risk Factor - A percentage risk of the Design not being completed according to the User Requirements (determined by what phases of the design Project have been completed along with the number of open items in the Open Item List)

Risk Factor routine - A routine of the Scenario package which determines the Risk Factor for the Project.

Running Task - A task in the Design Model which guides the User through a particular phase of the design process (e.g., Context Diagram) for a particular module of the system being designed.

Scenario package - A package in the Design Model which determines the different scenarios for the MES.

Scenario Database - A database which contains information about the different scenarios of the design Project.

Scenario Database Control task - A task in the Scenario package which controls entries into and retrievals from the Scenario Database.

User Interface - An interface of the MES between the User and the rest of the System.

User Interface Buffer - A task of the Kernel which buffers information coming from the Kernel to the User Interface.

Wait State - The condition that a task or process is in while it is waiting for something to happen (caused by another source).

Work Breakdown Structure - A method of structuring the total work of a project into a hierarchical set of sub-projects.

LIST OF REFERENCES

- 1 - B.A. Bowen and W.R. Brown, "Vol II of VLSI Systems Design for Digital Signal Processing", Prentice-Hall, Inc., 1985.
- 2 - R.J.A. Buhr, "Systems Design with ADA", Prentice-Hall, Inc., 1984.
- 3 - J.G. Cottrell, "Total Engineering: Control During Product Development", The Radio and Electronic Engineer, Vol. 49, No. 5, pp. 226-229, May 1979.
- 4 - I. Katz, "Automating Electronic Design", IEEE Spectrum, Vol 24, No. 5, May 1987.
- 5 - R. Davidson, "Product Planners Assess Users' Needs", IEEE Spectrum, Vol 24, No. 5, May 1987.
- 6 - P. Wallich, "How and When to Make Tradeoffs", IEEE Spectrum, Vol 24, No. 5, May 1987.
- 7 - N. Cortes-Comerer, "Motto for Specialists: Give Some, Get Some", IEEE Spectrum, Vol 24, No. 5, May 1987.
- 8 - D.E. Whitney, "Manufacturing and Design: a Symbiosis", IEEE Spectrum, Vol 24, No. 5, May 1987.
- 9 - V.G. Hajek, "Management of Engineering Projects", McGraw-Hill Book Co., 1977.

BIBLIOGRAPHY

Adams, J.R., Barndt, S.E., and Martin, M.D., "Managing by Project Management", Universal Technology Corporation, 1979.

Barnes, J.G.P., "Programming In ADA", Addison-Wesley Publishers Ltd., 1982.
- used as a guide for the pseudo-code in Appendix A

Bowen, B.A., and Brown, W.R., "Vol II of VLSI Systems Design for Digital Signal Processing", Prentice-Hall, Inc. 1985.
- used for the basic Methodology as presented in Chapter 2

Buhr, R.J.A., "Systems Design With ADA", Prentice-Hall, Inc., 1984.
- presents a method of illustrating a Design by using ADA

Cottrell, J.G., "Total Engineering: Control During Product Development", The Radio and Electronic Engineer, Vol. 49, No. 5, May 1979.

Chvátal, V., "Linear Programming", W.H. Freeman and Co., 1983.

Cortes-Comerer, N., 'Motto for Specialists: Give Some, Get Some', IEEE Spectrum, Vol 24, No. 5, May 1987.

Davidson, R., "Product Planners Assess Users' Needs", IEEE Spectrum, Vol 24, No. 5, May 1987.

Hajek, V.G., "Management of Engineering Projects", McGraw-Hill Book Co., 1977.

- recommended reading if the design of the Management Expert System should continue

Harvard Total Project Project Manager II ver 2.0, "User's Manual", Software Publishing Corp., 1986.

Katz, I., "Automating Electronic Design", IEEE Spectrum, Vol 24, No. 5, May 1987.

Shapiro, R.M., "Software for Designing Large Systems", IEEE Spectrum, Vol 24, No. 5, May 1987.

Wallich, P., "How and When to Make Tradeoffs", IEEE Spectrum, Vol 24, No. 5, May 1987.

Whitney, D.E., "Manufacturing and Design: a Symbiosis", IEEE Spectrum, Vol 24, No. 5, May 1987.

APPENDIX A

**PSEUDO-CODE FOR THE KERNEL
OF A MANAGEMENT EXPERT
SYSTEM**

Notes:

1. - This Appendix is meant for additional reading to support the Kernel's Structure Diagrams and to demonstrate their validity and logic.

2. - As there might be a difference in the way the pseudo-code is interpreted by different readers, the Design, as it is written in the main part of the text, should be referred to. Naming conventions are consistent with the rest of the Thesis to aid the reader in interpreting the pseudo-code.

3. - The pseudo-code was chosen to be based on a liberal interpretation of ADA because of ADA's ability to handle concurrent activities. In light of this, emphasis should be on the general thrust of the Design and not on the syntax of the pseudo-code, itself.

```
package Kernel is  
  type Mode_type is (Initialization,Daily_Startup,Continuous_Running);  
  Mode : Mode_type;  
  procedure Design (in Mode; inout other parameters and data  
                    necessary for the operation of the Kernel);  
  procedure Data_Input (in Mode, automatic input data);  
end Kernel;
```

```

package body Kernel is
  procedure Design (in Mode; inout other parameters and data necessary
                    for the operation of the Kernel);

    type Mode_type is (Initialization,Daily_Startup,Continuous_Running);
    Mode : Mode_type;

  begin
    case Mode is
      when Initialization => call Design_Model.Initialization (inout
        other parameters and data necessary for the
        operation of the Kernel);

      when Daily_Startup => call Design_Model.Daily_Startup (inout
        other parameters and data necessary for the
        operation of the Kernel);

      when Continuous_Running => call Design_Model.Continuous_
        Running (inout other parameters and data
        necessary for the operation of the Kernel);

      when others =>
        if command to end session then
          Data_Input.Stop (in command to end session);
          call Design_Model.Continuous_Running (in Mode; inout
            command to end session, status of it being
            stopped);
          Program_Management_Tools_Buffer.Stop (in command
            to end session);
          User_Interface_Buffer.Stop (in command to end session);
          - status that Design has stopped
        else
          - Error results
        end if;
    end case;
  end Design;

  procedure Data_Input (in Mode, automatic input data);
    type Mode_type is (Initialization,Daily_Startup,Continuous_Running);
    Mode : Mode_type;

  begin
    Data_Input.Enter (in Mode, automatic input data);      -- put data into
                                                              Data Input Buffer task
  end Data_Input;

package Design_Model is separate

```

```

begin
  task Data_Input is
    entry Enter (in Mode, automatic input data);
    entry Stop (in command to end session);
  end;

  task body Data_Input is
  begin
    loop
      select
        accept Enter (in Mode, automatic input data)
          call Design_Model.Data_Input (in Mode, other
            parameters and data necessary for the
            automatic entry of data);
        end;

      or

        accept Stop (in command to end session) end;
        - close off session
        exit loop;
      end select;
    end loop;
  end Data_Input;

  task User_Interface_Buffer is
    entry Enter (in data destined for the User Interface);
    entry Stop (in command to end session);
    Timeout : Boolean := 0;
    Data_n_buffer : Boolean := 0;
    Buffer_full : Boolean := 0;
  end;

  task body User_Interface_Buffer is
  begin
    loop
      select
        when not Timeout and not Buffer_full
          accept Enter (in data destined for the User Interface) do
            Data_in_buffer := 1;
            if buffer is full then Buffer_full := 1;
          end;
      end;
    end;
  end;

```

```

or when Data_in_buffer and not Timeout do
    MES.User_Interface (in data destined for the User
                        Interface);
    Data_in_buffer := 0;
    Buffer_full := 0;
end;

or when Timeout
    Timeout := 0;                -- reset Timeout flag

or
    accept Stop (in command to end session) end;
    if Data_in_buffer then
        MES.User_Interface (in data destined for the User
                        Interface);
    end if;
    - close off session
    exit loop;
end select;
end loop;
end User_Interface_Buffer;

task Program_Management_Interface is
    entry Enter (in data destined for the Program Management Tools),
    entry Retrieve (out data returning from the Program Management
                    Tools);
    entry Stop (in command to end session);
    Timeout : Boolean := 0;
    Data_Sent : Boolean := 0;
    Data_Returned : Boolean := 0;
end;

task body Program_Management_Interface is
begin
    loop
        select
            when not Timeout
                accept Enter (in data destined for the Program Management
                            Tools) end;

            or when Data_Returned and not Timeout
                accept Retrieve (out data returning from the Program
                            Management Tools);
        end select
    end loop
end

```

```

or when Timeout do
    Timeout := 0;           -- reset Timeout flag
    if data is in the buffer then
        while not Timeout loop
            MES.Program_Management_Tools (in data
                destined for the Program Management Tools);
            Data_Sent := 1;
        end loop;
    end if;

    if Data_Sent = 1 then
        while not Timeout loop
            MES.Program_Management_Tools (out data
                returning from Program Management Tools);
            Data_Sent := 0;
        end loop;
    end if;
    Timeout := 0;           -- reset Timeout flag
end;

or
    accept Stop (in command to end session) end;
    - close off session
    exit loop;
end select;
end loop;
end Program_Management_Interface;
end Kernel.

separate package Design_Model is
    type Mode_type is (Initialization,Daily_Startup,Continuous_Running);
    task type Running_Task is private;

    procedure Initialization (in Mode; inout other parameters and data
        necessary for the operation of the Kernel);
    procedure Daily_Startup (in Mode; inout other parameters and data
        necessary for the operation of the Kernel);
    procedure Continuous_Running (in Mode; inout other parameters and
        data necessary for the operation of the
        Kernel);

    procedure Data_Input (in Mode, other parameters and data necessary for
        the automatic entry of data);

    Mode : Mode_type;
end Design_Model;

```



```

package body Design_Model is
  procedure Initialization (in Mode; inout other parameters and data
                           necessary for the operation of the Kernel);

  begin
    if first time through then
      - make parameters necessary for initialization of modules
      call Scenario.Prospective (in Mode, parameters necessary for
                                initialization);
      Database_Control.Start (in Mode, parameters necessary for
                              initialization);
      Allocator/Scheduler.Initialization (in Mode, parameters necessary
                                          for initialization);
    else
      Allocator/Scheduler.Initialization (in parameters necessary for
                                          continuing Initialization mode);
    end if;
  end Initialization;

  procedure Daily_Startup (in Mode; inout other parameters and data
                           necessary for the operation of the Kernel);

  begin
    - make parameters necessary for initialization of modules for Daily
      Startup mode
    call Scenario.Prospective (in Mode, parameters necessary for Daily
                              Startup);
    Database_Control.Start (in Mode, parameters necessary for Daily
                            Startup);
    Allocator/Scheduler.Daily_Startup (in Mode, parameters necessary for
                                       Daily Startup);
  end Daily_Startup;

  procedure Continuous_Running (in Mode; inout other parameters and
                                data necessary for the operation of the
                                Kernel);
    Sub_mode : Mode_type;

  begin
    case Sub_mode is
      when Change Database =>
        if item is to be retrieved then
          - prepare the necessary parameters for retrieval
          Database_Control.Retrieve (in parameters; out data or
                                    questions);
        end if;
    end case;
  end Continuous_Running;

```

```

else if item is to be stored then
    - prepare the necessary parameters to put item in
      database
    Database_Control.Input (in parameters, out data or
      questions);

else
    - Error results
end if;

when Questions and Updates => do
    - prepare parameters necessary to get location of desired
      Running_Task
    Allocator/Scheduler.Continuous_Running (in Mode,
      parameters necessary to get location of
      desired Running_Task; out location of desired
      Running_Task);
    "Name".Command (in next Command, other data and
      parameters necessary for the execution of the
      Command);
end;

when Prospective Scenarios =>
    if new Prospective Scenario desired then
        call Scenario.Prospective (in Mode, command to create
          Prospective Scenario from Current);

    else if change to Prospective Scenario is desired then
        - prepare changes for Prospective Scenario
        call Scenario.Prospective (in Mode, command to change
          Prospective Scenario, changes);

    else if new Current Scenario is desired then
        call Scenario.Current (in Mode, command to create
          Current from Prospective and send status to
          User_Interface_Buffer);

    else
        - Error results
    end if;

```

```

when others =>
  if command to end session then
    - prepare parameters necessary to get location of all
      Running_Tasks
    Allocator/Scheduler.Stop (in Mode, parameters necessary
      to get location of all Running_Tasks; out
      location of all Running_Tasks, their names);

    while Running_Tasks are still left loop
      "Name".Stop (in command to end session);
    end loop;

    call Scenario.Stop (in command to end session);
    Database_Control.Stop (in command to end session);
    - status that the Design_Model has stopped

  else
    - Error results
  end if;
end case;
end Continuous_Running;

procedure Data_Input (in Mode, other parameters and data necessary for
  the automatic entry of data);
begin
  To be Determined
end Data_Input;

task Allocator/Scheduler is
  entry Initialization (in Mode, parameters necessary for initialization
    mode);
  entry Daily_Startup (in Mode, parameters necessary for Daily_Startup
    mode);
  entry Continuous_Running (in Mode, parameters necessary to get
    location of desired Running_Task; out
    location of desired Running_Task);
  entry Stop (in Mode, parameters necessary to get location of all
    Running_Tasks; out location of all Running_
    Tasks, their names);

  Timeout : Boolean := 0;
end;

```

```

task body Allocator/Scheduler is
begin
  loop
    select
      accept Initialization (in Mode, parameters necessary for
        initialization mode) end;
      - make parameters necessary for Initialization mode

      if first time through then
        - create parameters necessary for the creation of a
          Particular database
        Database_Control.Retrieve (in parameters necessary for
          the creation of a Particular database; out first
          set of questions);
        Kernel.User_Interface_Buffer.Enter (in next set of
          questions);
      else
        case next command is
          when get next set of questions => do
            - create parameters necessary to get next set of
              questions
            Database_Control.Retrieve (in parameters
              necessary to get next set of questions; out
              next set of questions);
            Kernel.User_Interface_Buffer.Enter (in next set of
              questions);
          end;

          when entry into Particular database => do
            - create parameters necessary to make entry into
              Particular database
            Database_Control.Input (in parameters necessary
              to make entry into Particular database, data to
              be entered);
            Kernel.User_Interface_Buffer.Enter (in resulting
              change in Particular database);
          end;

          when create Prospective Scenario => call Scenario.
            Prospective (in Mode, command to create
              new Prospective Scenario);
        end;
      end
    end
  end

```

3 OF/DE 3

MicroD

```

        when change Prospective Scenario => do
            - prepare changes for Prospective Scenario
            call Scenario.Prospective (in Mode, command to
                change Prospective Scenario, changes);
        end;
    end case;
end if;

```

or

```

    accept Daily_Startup (in Mode, parameters necessary for
        Daily_Startup mode) end;
    - make parameters necessary for Daily Startup mode
    call Scenario.Current (in Mode, command to create Current
        from Prospective and return status; out list of
        Running Tasks, their names, types and status);

```

```

    while no more Running Tasks loop
        "Name" := name of next Running Task;
        "Name" : Running Task;                -- activate "name"
        "Name".Start (in type, status, other parameters necessary
            for its operation);
    end loop;

```

or when not Timeout

```

    accept Continuous_Running (in Mode, parameters necessary
        to get location of desired Running_Task; out
        location of desired Running_Task) do
        - find location of desired Running_Task
        if it does not exist then
            call Scenario.Current (in Mode, command to return
                the latest Current Scenario status; out list of
                Running Tasks, their names, types and status);
            if it still does not exist then
                - Error results
            end if;
        end if;
    end if;
end;

```

or when Timeout do

-- This allows the Allocator/Scheduler task to periodically update its internal status of the Running_Tasks that are active or are supposed to be active. If there are new Running_Tasks that are supposed to be active, the Allocator/Scheduler task would activate them.

```

    - make parameters necessary to get latest Current
      Scenario

```

```

    call Scenario.Current (in Mode, command to return the
                          latest Current Scenario status; out list of
                          Running Tasks, their names, types and status);
    - compare latest list with the previous
    if names on latest list were not on previous then
        -list them as outstanding
        while no more Running Tasks outstanding loop
            "Name" := name of next Running Task;
            "Name" : Running Task;          -- activate "name"
            "Name".Start (in type, status, other parameters
                          necessary for its operation);
        end loop;
        - make latest list the "current" reference
    end if;
    Timeout := 0;
end;

or

accept Stop (in Mode, parameters necessary to get location
            of all Running_Tasks; out location of all
            Running_Tasks, their names)
call Scenario.Current (in Mode, command to return the
                      latest Current Scenario status; out list of
                      Running Tasks, their names, types and status)

end;
exit loop;
end select;
end loop;
end Allocator/Scheduler;

task Database_Control is
    entry Start (in Mode, parameters necessary for initialization);
    entry Input (in parameters necessary to make entry into desired
                database, data to be entered);
    entry Retrieve (in parameters; out data or questions);
    entry Stop (in command to end session);
end;

task body Database_Control is
begin
    loop
        select
            -- This will allow the Database to be able to prepare for the
            Initialization and Daily_Startup modes when the MES is first turned on or for the
            switchover between the two modes

```

accept Start (in Mode, parameters necessary for operation of the Database)

- determine which databases are available to the MES

end;

or

accept Input (in parameters necessary to make entry into desired database, data to be entered) **do**

- open the database desired
- enter data into the proper place in the database
- close the database

end;

or

accept Retrieve (in parameters; **out** data or questions)

case command is

when create a new Particular Database => **do**

- create the Particular database desired
- open the Generic database
- copy Generic set of references into Particular database
- get the next set of references
- close Particular database
- get the next set of questions
- close the Generic database

end;

when get next set of questions => **do**

- open the Particular database desired
- get the next set of references
- close the Particular database
- open the Generic database
- get the next set of questions
- close the Generic database

end;

when get associated risk factors of unanswered questions => **do**

- open the Generic database desired
- get the risk factors associated with the references
- close the Generic database

end;

end case;

end;


```

        or
            accept Stop (in command to end session)
                - close any open databases
            end;
            exit loop;
        end select;
    end loop;
end Database_Control;

task type Running_Task is
    entry Start (in type, status, other parameters necessary for its
                operation);
    entry Command (in next Command, other data and parameters
                necessary for the execution of the Command);
    entry Stop (in command to end session);
end;

task body Running_Task is
begin
    accept Start (in type, status, other parameters necessary for its
                operation) end;
    - set up the necessary parameters for the continuous running of this
      task
    Database_Control.Retrieve (in parameters to create new Particular
                database; out data or questions)
    call Scenario.Current (in Mode; inout other parameters and data
                necessary to get open items assigned to this
                task);

    loop
        select
            accept Command (in next Command, other data and
                parameters necessary for the execution of the
                Command) end;

            case next command is
                when answer a question => do

                    -- Since each line of questioning has a dependence/precedence
                    relationship, the answers of some questions will lead to further questions or to
                    the continuing of the next module.

                    Database_Control.Input (in parameters necessary to
                        make entry into Particular database, answer to
                        be entered)
                    - determine next question based on answer
                    - prepare parameters to update current status

```

```

    call Scenario.Current (In Mode; inout other
                          parameters and data necessary to update the
                          status of this task);
    if this task is finished then exit loop;
end;

when request to continue => do

```

-- The Project Manager may wish to continue to the next module while leaving some unanswered questions. In this case, the Running_Task must determine if there is sufficient data to allow the next phase of the design process to continue. It then presents the User with the associated risk factor if he/she wishes to proceed.

```

    - determine the unanswered questions
    Database_Control.Retrieve (in references to questions
                              in order to determine risks; out associated
                              risks, maximum risk allowed to continue past
                              this phase)
    - add up risks

    if total risk < max. allowed then
        Kernel.User_Interface_Buffer.Enter (in associated
                                             risk, maximum allowed)
        - add all questions to an open item list
        call Scenario.Current ('n Mode, all questions to
                              be added to the open item list; inout other
                              parameters and data necessary to send stop
                              status);
        exit loop;

    else
        Kernel.User_Interface_Buffer.Enter (in associated
                                             risk, maximum allowed, "not allowed - too
                                             much risk")
    end if;
end;

when split up Task =>
    if first time through then
        - prepare parameters necessary for splitting tasks
        Database_Control.Retrieve (in parameters to get
                                    Generic questions for splitting tasks; out
                                    questions)
        Kernel.User_Interface_Buffer.Enter (in Generic
                                             questions for splitting tasks)
    end if;
end;

```

```

else
  - from the answers, determine where the logical
    division of the task should take place
    (between mutually exclusive sub-modules of
    the Design)
  - prepare parameters to split up the task and
    divide the open items
  call Scenario.Prospective (in Mode, parameters
    necessary to create a split)
  call Scenario.Current (in Mode, all questions to
    be added to the Open Item lists of the next
    tasks; inout other parameters and data
    necessary to send stop status);
  exit loop;
end if;
end;

when update current status => do
  - prepare parameters to update current status
  call Scenario.Current (in Mode; inout other
    parameters and data necessary to update the
    status of this Running_Task);
  if this task is finished then exit loop;
end;

when get next set of questions => do
  Database_Control.Retrieve (in get next set of
    references to questions from Particular
    database; out references)
  - determine parameters necessary to get next set of
    questions
  Database_Control.Retrieve (in parameters to get next
    set of questions; out questions)
  Kernel.User_Interface_Buffer.Enter (in next set of
    questions)
end;

when others => Error results;
end case;

```

```

    or
        accept Stop (in command to end session) do
            Database_Control.Input (in parameters necessary to make
                entry into Particular database, closing status);
            call Scenario.Stop (in closing status);
        end;
        exit loop;
    end select;
end loop;
end Running_Task;

package Scenario is
    procedure Prospective (in Mode, other parameters and data
        necessary for the Prospective Scenario);
    procedure Current (in Mode; inout other parameters and data
        necessary for the Current Scenario);
    procedure Stop (in command to end session);
    type Tag is private;
end Scenario;

package body Scenario is
    type Tag is
        record
            Date : Integer := 0
            Time : Integer := 0
        end record;

    procedure Prospective (in Mode, other parameters and data
        necessary for the Prospective Scenario);
    Date_Time : Tag

begin
    case command is
        when create new Prospective Scenario => do
            - prepare parameters necessary to create new Prospective
              Scenario
            if first time through then
                Scenario_Database_Control.Start (in Mode,
                    parameters necessary for the operation of the
                    Scenario Databases);
            end if;
            Prospective.New (in Mode, other parameters and data
                necessary for the Prospective Scenario);
        end;
    end;
end;

```

```

when create Prospective from Current => do
  - prepare parameters necessary to create Prospective
    Scenario from Current
  if Mode = Daily_Startup then
    Scenario_Database_Control.Start (in Mode,
      parameters necessary for the operation of the
      Scenario Databases);
  end if;
  Prospective.Revise (in Mode, other parameters and data
    necessary for the Prospective Scenario);
end;

when change Prospective Scenario => do
  - prepare parameters necessary to change the Prospective
    Scenario
  Prospective.Revise (in Mode, other parameters and data
    necessary for the Prospective Scenario);
end;

when split tasks => do
  - prepare parameters necessary to split tasks
  if antecedents for new tasks have not been met then
    - Error results
  end if;
  Prospective.Split (in Mode, other parameters necessary to
    split tasks; out Status);
  Prospective.Split (in Mode; out Status);
  Scenario_Database_Control.Copy (in parameters
    necessary to copy data from Prospective
    Database to Current; out Date_Time);
  Current_Date_Time := Date_Time;
  Scenario_Database_Control.Retrieve (in parameters
    necessary to get Current Scenario; out
    Current Scenario);
  - determine which Open Items and Running Tasks are on
    the critical path and flag them
  Kernel.User_Interface_Buffer.Enter (in Current Scenario);
end;

when others => Error results
end case;
end Prospective;

```

```

procedure Current (in Mode; inout other parameters and data
                    necessary for the Current Scenario);
Current_Date_Time : Tag
Date_Time : Tag

Begin
  case command is
    when make Current from Prospective => do
      Scenario_Database_Control.Retrieve (in parameters to
        get date/time tag of the Prospective Scenario;
        out Date_Time);

      if Current_Date_Time > Date_Time then
        Kernel.User_Interface_Buffer.Enter (in flag to indicate
          that the Current Scenario had changed since
          the Prospective Scenario analysis had
          started);
      else
        Scenario_Database_Control.Copy (in parameters
          necessary to copy data from Prospective
          Database to Current; out Date_Time);
        Current_Date_Time := Date_Time;
        Scenario_Database_Control.Retrieve (in parameters
          necessary to get Current Scenario; out
          Current Scenario);
        - determine which Open Items and Running Tasks are
          on the critical path and flag them
        Kernel.User_Interface_Buffer.Enter (in Current
          Scenario);
      end if;
    end;

    when make Current from Prospective anyway => do
      Scenario_Database_Control.Copy (in parameters
        necessary to copy data from Prospective
        Database to Current; out Date_Time);
      Current_Date_Time := Date_Time;
      Scenario_Database_Control.Retrieve (in parameters
        necessary to get Current Scenario; out
        Current Scenario);
      - determine which Open Items and Running Tasks are on
        the critical path and flag them
      Kernel.User_Interface_Buffer.Enter (in Current Scenario),
    end;

```

```

when update status of Current Scenario => do
    - get the current date and time from the system
    Current_Date_Time := the current date and time
    Scenario_Database_Control.Input (in parameters
        necessary to input new Current Scenario
        status, Project data to be entered);
    call Cost_Factor (in Scenario to be reviewed);
    call Risk_Factor (in Scenario to be reviewed);
    Scenario_Database_Control.Retrieve (in parameters; out
        Project Scenario data);
    - determine which Open Items and Running Tasks are on
        the critical path and flag them
    Kernel.User_Interface_Buffer.Enter (in Project data
        destined for the User Interface);
end;

when get Current Scenario => do
    Scenario_Database_Control.Retrieve (in parameters; out
        Project Scenario data);
    if Scenario is to be sent to the User_Interface then
        - determine which Open Items and Running Tasks are
            on the critical path and flag them
        Kernel.User_Interface_Buffer.Enter (in Project data
            destined for the User Interface);
    end if;
end;
end case;
end Current;

procedure Stop (in command to end session or to save closing
    status);
begin
    if command save closing status then
        Scenario_Database_Control.Input (in parameters necessary
            to make entry into Current Database, data to
            be entered)
    else
        Prospective.Stop (in command to end session);
        Scenario_Database_Control.Stop (in command to end
            session);
        -close off session
    end if;
end Stop;

```

procedure Risk_Factor (In Scenario to be reviewed);

Total_Risk : real := 0

begin

**Scenario_Database_Control.Retrieve (In parameters to get
desired scenario; out desired scenario,
particular project data);**

- determine where the next milestone is in the Project

if Project is at the milestone (within the allowable tolerance) then

if there has been a Design review then

**- determine which tasks have proceeded past the
milestone "at risk"**

- determine which tasks have not reached the milestone

- determine open items as a result of the review

- determine other open items

**Total_Risk := Project risk (for the following milestone) + risk
of tasks proceeding past the milestone + risk
of the tasks yet to reach the milestone + risk of
all open items**

else

- raise a flag to have a Design review

**- determine which tasks have proceeded past the
milestone "at risk"**

- determine which tasks have not reached the milestone

- determine other open items

**Total_Risk := Project risk (for the milestone) + risk of tasks
proceeding past the milestone + risk of the
tasks yet to reach the milestone + risk of all
open items**

end if;

else

- determine which tasks have not reached the last milestone

- determine other open items

**if there exists tasks which have not reached the last milestone
or there are other open items related to the
milestone then**

**- determine which tasks have proceeded past the last
milestone "at risk"**

**Total_Risk := Project risk (for the milestone concerned) +
risk of tasks proceeding past the milestone +
risk of the tasks yet to reach the milestone +
risk of other open items**


```

        else
            Total_Risk := Project risk (for the next milestone) + risk of
                           open items + risk of tasks which have not yet
                           reached the next milestone
        end if;
    end if;
    Scenario_Database_Control.Input (in parameters necessary to
                                     enter Total_Risk and Design review flag (if
                                     any) into desired database, Total_Risk,
                                     Design review flag)
end Risk_Factor;

procedure Cost_Factor (in Scenario to be reviewed);
    Total_Cost : real := 0

begin
    Scenario_Database_Control.Retrieve (in parameters to get
                                         desired scenario; out desired scenario,
                                         particular project data);
    - determine the actual costs already incurred
    - based on the particular project data, determine the projected cost
      until the end of the Project
    Total_Cost := Actual costs + projected costs
    Scenario_Database_Control.Input (in parameters necessary to
                                     enter Total_Cost into desired database,
                                     Total_Cost)
end Cost_Factor;

begin
    task Prospective is
        entry New (in Mode, other parameters and data necessary for the
                  Prospective Scenario);
        entry Revise (in Mode, other parameters and data necessary for
                     the Prospective Scenario);
        entry Split (in Mode, other parameters necessary to split tasks;
                     out Status);
        entry Stop (in command to end session);
        Date_Time : Tag;
        Task_Split : Boolean := 0;
        Splitting_in_Progress : Boolean := 0;
end;

```

```

task body Prospective is
begin
  loop
    select
      accept New (in Mode, other parameters and data
                  necessary for the Prospective Scenario) end;
    if not first time through then
      Scenario_Database_Control.Delete (in reference to
                                         Prospective file to be deleted);
    end if;
    Design_Model.Database_Control.Retrieve (in parameters;
                                             out Project data in Particular database);
    - using the particular Project data, organize it to make
      Project tasks for each phase allowing for the
      Design's hierarchical structure
    - determine the resources available for the Project
    - modify all data to satisfy Program Management Tools'
      requirements
    Kernel.Program_Management_Interface.Enter (in Project
                                                data destined for the Program Management
                                                Tools);
    Scenario_Database_Control.Input (in parameters
                                     necessary to create new Prospective
                                     Database, Project data to be entered);
    Kernel.Program_Management_Interface.Retrieve (out
                                                  data returning from the Program Management
                                                  Tools);
    Scenario_Database_Control.input (in parameters
                                     necessary to input new Prospective Scenario
                                     data, Project data to be entered);
    call Cost_Factor (in Scenario to be reviewed);
    call Risk_Factor (in Scenario to be reviewed);
    Scenario_Database_Control.Retrieve (in parameters; out
                                         Project Scenario data);
    Kernel.User_Interface_Buffer.Enter (in Project data
                                         destined for the User Interface);

  or

    accept Revise (in Mode, other parameters and data
                  necessary for the Prospective Scenario) end;

    if first time through then
      Design_Model.Database_Control.Retrieve (in
                                              parameters; out Project data in Particular
                                              database if changed since Daily_Startup);
      - prepare parameters to make Prospective Scenario

```

Scenario_Database_Control.Copy (in parameters
necessary to copy data from Current Database
to Prospective Database; out Date_Time);

if Mode $\neq$ Daily_Startup **then**

Kernel.User_Interface_Buffer.Enter (in Project
data destined for the User Interface);

else

- modify data to satisfy Program Management
Tools' requirements

Kernel.Program_Management_Interface. Enter (in
data destined for the Program Management
Tools);

Kernel.Program_Management_Interface. Retrieve
(out data returning from the Program
Management Tools);

Scenario_Database_Control.Input (in parameters
necessary to input Prospective Scenario data,
Project data to be entered);

call Cost_Factor (in Scenario to be reviewed);

call Risk_Factor (in Scenario to be reviewed);

Scenario_Database_Control.Retrieve (in
parameters; out Project Scenario data);

Kernel.User_Interface_Buffer.Enter (in Project
data destined for the User Interface);

end if;

else

- modify data to satisfy design process flow and
Program Management Tools' requirements

Kernel.Program_Management_Interface. Enter (in
data destined for the Program Management
Tools);

Kernel.Program_Management_Interface. Retrieve (out
data returning from the Program Management
Tools);

Scenario_Database_Control.Input (in parameters
necessary to input Prospective Scenario data,
Project data to be entered);

call Cost_Factor (in Scenario to be reviewed);

call Risk_Factor (in Scenario to be reviewed);

Scenario_Database_Control.Retrieve (in parameters;
out Project Scenario data);

Kernel.User_Interface_Buffer.Enter (in Project data
destined for the User Interface);

end if;

```

or when not Splitting_in_Progress do
  accept Split (in Mode, other parameters necessary to split
               tasks; out Status)
  if Task_Split then
    Status := "Task Split";
  else
    Status := "Splitting in Progress";
    Splitting_in_Progress := 1;
  end if;
end;

if Splitting_in_Progress then
  Design_Model.Database_Control.Retrieve (in
    parameters; out Project data in Particular
    database if changed since Daily_Startup);
  - prepare parameters to make Prospective Scenario
  Scenario_Database_Control.Copy (in parameters
    necessary to copy data from Current Database
    to Prospective Database; out Date_Time);
  - identify task to be split
  - transfer open items to the proper Open Item lists
  - identify the necessary tasks to follow in accordance
    with the process flow model
  - modify data to satisfy Program Management Tools'
    requirements
  Kernel.Program_Management_Interface. Enter (in
    data destined for the Program Management
    Tools);
  Kernel.Program_Management_Interface. Retrieve (out
    data returning from the Program Management
    Tools);
  Scenario_Database_Control.Input (in parameters
    necessary to input Prospective Scenario data,
    Project data to be entered);
  call Cost_Factor (in Scenario to be reviewed);
  call Risk_Factor (in Scenario to be reviewed);
  Scenario_Database_Control.Retrieve (in parameters;
    out Project Scenario data);
  Splitting_in_Progress := 0;
end if;

```

```

        or
            accept Stop (in command to end session)
                - close off session
            end;
            exit loop;
        end select;
    end loop;
end Prospective;

task Scenario_Database_Control is
    entry Start (in Mode, parameters necessary for the operation of
        the Scenario Databases);
    entry Input (in parameters necessary to make entry into desired
        database, data to be entered);
    entry Retrieve (in parameters; out data);
    entry Copy (in parameters necessary to copy data from one
        Database to another; out Date_Time);
    entry Delete (in reference to Prospective file to be deleted);
    entry Stop (in command to end session);
    Date_Time : Tag
end;

task body Scenario_Database_Control is
begin
    loop
        select
            accept Start (in Mode, parameters necessary for
                operation of the Scenario Databases) do
                - determine which databases are available to this
                    Project
                - get the current date and time from the system
                Date_Time := the current date and time
            end;

        or

            accept Input (in parameters necessary to make entry into
                desired database, data to be entered)
            case command is
                when create new Prospective Database => do
                    - create new Prospective Database
                    - enter data into the proper place in the
                        Database
                    - close the Database
                end;
            end;
        end select;
    end loop;
end;

```

```

        when input data into Database => do
            - open the Database desired
            - enter data into the proper place in the
              Database
            - close the Database
        end;

        when others => Error results;
    end case;
end;

or

accept Retrieve (in parameters; out data)
case command is
    when retrieve Prospective Scenario => do
        - open Prospective Scenario Database
        - get the Prospective Scenario data
        - close Prospective Scenario Database
    end;

    when retrieve Current Scenario status => do
        - open Current Scenario Database
        - get the Current Scenario status
        - close Current Scenario Database
    end;

    when others => Error results
end case;
end;

or

accept Copy (in parameters necessary to copy data from
             one Database to another; out Date_Time) do
    - get the current date and time from the system
    Date_Time := the current date and time

case command is
    when copy from Current to Prospective =>
        - copy Current Scenario Data and Date_Time
          into Prospective Database

```

```

        when copy from Prospective to Current =>
            - copy Prospective Scenario Data and
              Date_Time into Current Database

        when others => Error results
    end case;
end;

or
    accept Delete (in reference to Prospective file to be
                  deleted);
    - delete Prospective file
end;

or
    accept Stop (in command to end session)
    - close any open databases
    end;
    exit loop;
end select;
end loop;
end Scenario_Database_Control;
end Scenario;
end Design_Model.

```

APPENDIX B

TYPICAL LOGIC FLOW

THROUGH THE PHASES OF

A DESIGN PROCESS

This Appendix includes some typical questions and processes for the Generic part of the Design Database. These cover each of the design phases and would be used by the Running Tasks in the Continuous Running mode of the MES. Each section (Context Diagram, Data Flow Analysis, etc.) has three parts:

1. - Antecedents - the list of things which must exist before the Running Task can proceed;

2. - Process - the list of questions and Project Status Updates which provide sequencing and direction to the Project Manager;

and,

3. Consequences - the list of results if the Running Task were to be completed.

The process flow and questions given are meant as a guide for future development of the MES. Some processes are illustrated in the form of loops for clarity. In practice, there would be a series of links between entries to

perform the actual sequencing. It is acknowledged that other parts of the Generic Database would have to be added (e.g., Risk Factors for Open Items, flags for required re-design, etc.) and that the Physical Design and Testing sections would have to be further broken out.

The occurrence of **"re-design"** means that a flag would be sent to the User which will require a change to the Project Scenario because of the requirement for re-design.

Context Diagram

Antecedents

- User Requirements

Process

- check the User Requirements for
 - functional requirements
 - non-functional requirements
 - user visible functions
 - user-defined interactions between different sections of the system
- is all the information necessary to start the design present?
 - no - is there enough to start some of it?
 - yes - continue at "risk"
 - call Customer and get the required information
 - re - check the User Requirements (i.e., start over)
 - yes - continue

loop

- group all the User visible functions into separate modes and label each mode according to its primary function (e.g., operator interface)
- establish the interactions between each mode and the other functional requirements not visible to the user
- create an onion skin model (Context Diagram) of the system showing the interactions between each layer and between each mode
- check the Context Diagram against the Functional Requirements, User visible functions and Interactions
- does the Context Diagram reflect the system adequately?
 - yes - exit loop
 - no - determine why
 - should not proceed "at risk" beyond this point because too much risk for re-design
- call Customer if necessary to get more information

end loop;

- conduct a Formal Review with Customer
- does the Context Diagram reflect the system adequately enough to proceed?
 - no - determine why and start loop again
 - yes - any outstanding issues?
 - yes - put them on the Open Item List
 - clean up paperwork for this phase
 - can a split be made between two mutually exclusive modes?
 - no - continue with next phase without splitting
 - yes - make the desired split
 - name the parts
 - continue with next phase

Consequences

- Context Diagram

Next tasks

- Data Flow Analysis
- names

Data Flow Analysis

Antecedents

- User Requirements
- Context Diagram

Process

- are the required antecedents for this task in place?
 - no - wait until they are in place
- place the system functions in a network such that data flows from left to right
 - (some functions may have to be divided in order to keep the "left to right" rule in tact)

loop

- subdivide each function in a pipeline fashion such that no further divisions are possible
- subdivide each function in a parallel manner in order to achieve maximum parallelism
- check each module to see if it has sufficient granularity to be able to determine execution times and data flow rates
- is there sufficient granularity?
 - yes - exit loop

end loop;

- is the Data Flow Graph reflective of the functioning of the system?
 - no - start loop again
- create a Data Dictionary by listing in tabular form all the edges in the system along with their data flow rates and then all the nodes in the system along with their data transformation rates

loop

- determine all the data transfer rates to the inputs of the first nodes (or next set of nodes in the following times through the loop)
- determine the minimum execution time for the data transformations in the nodes. If all the input data is not ready for a particular node, then skip this step for that node
- determine the maximum output rates to the next nodes (or the output of the system if a node in the last set)
- any more nodes remaining in the system?
 - no - exit loop

end loop;

- check the system output data transfer rates against the User Requirements
- do the rates do meet the requirements?
 - no - call Customer to change requirements if possible
 - **Re-design** - make changes as necessary
- determine where data transfer rates or data transformations could be slowed down without reducing overall system speed but to decrease costs
- determine where parallelism could be reduced (again to reduce cost) without reducing performance
- organize the Loading Analysis (graphs etc.)
- conduct a Formal Review with Customer and other interested parties within the Company (e.g., Quality Assurance, Production, Logistics Support, etc.)
- does the Data Flow Analysis reflect the system adequately enough to proceed?
 - no - determine why and start loop again
 - yes - any outstanding issues?
 - yes - put them on the Open Item List
 - any outstanding items from previous phases?
 - yes - do not continue until they are cleared up
 - clean up paperwork for this phase
 - can a split be made between two mutually exclusive modes?
 - no - continue with next phase without splitting
 - yes - make the desired split
 - name the parts
 - continue with next phase

Consequences

- Data Dictionary
- Data Flow Graph
- Loading Analysis

Next tasks

- Structure Diagrams
- names

Structure Diagrams

Antecedents

- User Requirements
- Context Diagram
- Data Dictionary
- Data Flow Graph
- Loading Analysis
- Next higher level Structure Diagram (if it exists)

Process

- are the required antecedents for this task in place?

no - wait until they are in place

loop

if this is the first time through the loop then

- use the next higher level Structure Diagram (if it exists) or the Context Diagram and Data Flow Graph as the basis for design

else

- use the next higher level Structure Diagram and Data Flow Graph as the basis for design

end if;

- establish the mutually exclusive modes, functions and input/ output data
- group the modes/functions into packages
- using the following guidelines, create a Structure Diagram:
 - tasks
 - target task, low utilization
 - configure target as server
 - target task, high utilization
 - configure target as actor
 - many senders, either target busy or many targets
 - use buffer task as server

- passive packages or limited resources (e.g., data bus, memory disk, etc.)
 - may require an allocator or arbiter task
- does the Structure Diagram have sufficient detail to start the Physical Design?
 - yes - exit loop
 - no - go to the next lower level
- end loop;
- determine the additional timing considerations due to the control aspect of the design
- determine the necessary data transferred for control purposes
- make the necessary changes to the Loading Analysis and Data Dictionary
- compare the Loading Analysis with the User Requirements
- does the Design does meet the User Requirements?
 - no - determine the changes necessary
 - do changes require customer approval?
 - yes - call Customer
 - **re-design** to incorporate the changes as necessary (this may require changing the Context Diagram, Data Flow Graph, Data Dictionary, Loading Analysis and/or Structure Diagrams)
- conduct a Formal Review with Customer and other interested parties within the Company (e.g., Quality Assurance, Production, Logistics Support, etc.)
- do the Structure Diagrams reflect the system adequately enough to proceed?
 - no - determine why and start loop again
 - yes - any outstanding issues?
 - yes - put them on the Open Item List
 - any outstanding items from previous phases?
 - yes - do not continue until they are cleared up
 - clean up paperwork for this phase
 - can a split be made between two mutually exclusive modes?
 - no - continue with next phase without splitting
 - yes - make the desired split
 - name the parts
 - continue with next phase

Consequences

- Structure Diagram
- Next tasks
 - Physical Designs
 - names

Physical Design

Antecedents

- User Requirements
- Context Diagram
- Data Dictionary
- Data Flow Graph
- Loading Analysis
- Structure Diagrams

Process

- are the required antecedents for this task in place?
 - no - wait until they are in place
- technology review
- can the current state of the Design be met with the current technology?
 - no - determine changes necessary in order to be able to continue
 - call Customer with discrepancies
 - still continue with the Design?
 - no - wait until technology catches up
 - yes - **re-design** to make the changes necessary in order to continue
- evaluate the functional elements versus technology and Company capabilities
- loop
 - select functional element
 - does Design already exist?
 - yes - use existing design?
 - yes - use the Design
 - no - partition the algorithm
 - allocate the parts to hardware or software
 - split the task?
 - yes - conduct splits necessary
 - no - research technology if necessary to determine what is necessary for the Design
 - continue with Physical Design
 - is the Design complete to the lowest level?
 - yes - exit loop
 - no - go to next lower level of Design
 - split the task?
 - yes - conduct splits necessary
- end loop;

- does the Design meet the User Requirements?
 - no - determine the changes necessary
 - do changes require customer approval?
 - yes - call Customer
 - **re-design** to incorporate the changes as necessary (this may require changing the Context Diagram, Data Flow Graph, Data Dictionary, Loading Analysis, Structure Diagrams and/or other Physical Designs which are complete
- conduct a Formal Review with Customer, and other interested parties within the Company (e.g., Quality Assurance, Production, Logistics Support, etc.)
- does the Physical Design reflect the system adequately enough to proceed?
 - no - determine why and start loop again
 - yes - any outstanding issues?
 - yes - put them on the Open Item List
 - any outstanding items from previous phases?
 - yes - do not continue until they are cleared up
 - clean up paperwork for this phase
 - can a split be made between two mutually exclusive modes?
 - no - continue with next phase without splitting
 - yes - make the desired split
 - name the parts
 - continue with next phase

Consequences

- Physical Design
- Next tasks
- Testing
 - names

Testing

Antecedents

- User Requirements
- Context Diagram
- Data Dictionary
- Data Flow Graph
- Loading Analysis
- Structure Diagrams
- Physical Design
- successful test results from the next lower levels (if there are lower levels)

Process

- are the required antecedents for this task in place?
 - no - wait until they are in place
- are there any outstanding items?
 - yes - do not continue until they are cleared
- test the Design and compare the results to the User Requirements for this level
- does the Design meet the User Requirements?
 - no - determine why it does not
 - retest if required
 - call customer to discuss the necessary changes in order to meet the requirements
 - **re-design** to incorporate the changes as necessary (this may require changing the Context Diagram, Data Flow Graph, Data Dictionary, Loading Analysis, Structure Diagrams and/or other Physical Designs which are complete
 - retest up to this level
- is this the top level?
 - yes - conduct Formal Review with the Customer
 - acceptance of the Design

Consequences

- fully tested and working Design to this level

Next tasks

- Testing
 - names of the tasks for next higher level

END

21|09|90

FIN