Development and Verification of Engineering Design Iteration Models

by

Robert Paul Smith

B.S., Leland Stanford Junior University (1986)

S.M., Massachusetts Institute of Technology (1988)

Submitted to the Sloan School of Management in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the Massachusetts Institute of Technology

August 1992

© Massachusetts Institute of Technology

Signature of Author ____________________________

Sloan School of Management

August 7, 1992

Certified by ____________________________

Steven D. Eppinger

Assistant Professor of Management Science

Thesis Supervisor

Accepted by ____________________________

James B. Orlin

Chair, Doctoral Program Committee

SEP 18 1992
Development and Verification of Engineering Design Iteration Models

by Robert P. Smith

Submitted to the Sloan School of Management in partial fulfillment of the requirements for the degree of Doctor of Philosophy

This thesis develops several models of the engineering design iteration process, and attempts to verify the models in multiple ways. As extensions of the design structure matrix method, the models are based on mapping the input-output flows of information between discrete tasks in a design process. The information flows include cycles which preclude the use of a purely sequential process. The models are of two basic types, the Sequential Iteration probabilistic type, and the Work Transformation type. Each type of model also includes variations.

The Sequential Iteration model can suggest ways that an engineering design iteration cycle can be resequenced in order to lessen expected development time. The Sequential Iteration model assumes that the need for iteration is a probabilistic choice, and the order that the tasks are attempted affects the expected length of time to complete the design process.

The Work Transformation model can show which portions of the design problem cause the bulk of time during the iteration process. The Work Transformation model is a discrete time dynamic system description of iteration, where the amount of time taken to complete each iteration stage is a linear function of the amount of work done in the previous iteration stage. The Work Transformation model is verified in two ways: by comparing model predictions with industrial design practice, and by demonstrating model predictions in a controlled experimental setting.

Two industrial design settings are considered in the verification process. These are brake system design at General Motors, and electronics module design at Digital Equipment. These settings illustrate that the Work Transformation model can describe the design process and reveal system level interactions which control development time.

The Delta Design Game serves as the test bed for a controlled experiment which demonstrates that the Work Transformation model can compare different design strategies. For this game, two different design strategies were identified, and a Work Transformation Matrix was constructed for each strategy, predicting different completion times for each. Experiments were then conducted which demonstrated that the strategy predicted to be faster was indeed so, while the final designs were of comparable overall quality.

Thesis Supervisor: Dr. Steven D. Eppinger
Assistant Professor of Management Science
Acknowledgment

Support for this research has been generously provided by the Leaders for Manufacturing program. I have been pleased to see its development. It is the right thing at the right time. The program provides a good model for interdisciplinary research in engineering and management. I feel very fortunate to have conducted my graduate study during the rise of this program.

I wish to thank those at General Motors and Digital Equipment who gave generously of their time to assist me in my work, and particularly, to thank Gus Tayeh and Steve Denker for their help.

I thank my doctoral thesis committee, who have offered useful and practical guidance.

I wish to thank the graduate students who have made Sloan a pleasant place to be over the last four years. Also thanks to the Sloan administrative staff.

I very much thank Steven Eppinger for his excellent support as my advisor over the last four years. I will model my interactions with students after Steven's fine example.

Thanks to my mom. I couldn't have done it without you.

Thanks to Bert. It is a good thing to have your brother a few blocks away for six years. And also good to have your sister nearby.

Thanks to Tali. May the next sixty-two years be good ones.

A man said to the universe:
"Sir, I exist!"
"However," replied the universe,
"The fact has not created in me
A sense of obligation."

Steven Crane
Table of Contents

Abstract .......................................................................................................................... 2
Acknowledgements ........................................................................................................... 3
Table of Contents ............................................................................................................. 4

Chapter 1. Introduction .................................................................................................. 6
  1.1. Overview of Engineering Design ........................................................................... 6
  1.2. The Design Process ............................................................................................... 7
  1.3. Iteration in Engineering Design ............................................................................ 9
  1.4. Existing Models for Analyzing the Design Process ................................................ 10
    1.4.1. Information Flow Models ............................................................................. 11
    1.4.2. Time-Based Models ..................................................................................... 19
  1.5. What this Thesis Does ......................................................................................... 20

Chapter 2. Design Structure Matrices .......................................................................... 22
  2.1. History .................................................................................................................. 22
  2.2. Partitioning Design Structure Matrices ................................................................ 24
  2.3. Other Theoretical Issues ...................................................................................... 26
    2.3.1. Shunt Diagrams .......................................................................................... 27
    2.3.2. Redefining Tasks ......................................................................................... 28
  2.4. Applications of the DSM Method ......................................................................... 29
    2.4.1. Black ........................................................................................................... 29
    2.4.2. NASA ......................................................................................................... 30
    2.4.3. Sequeira ...................................................................................................... 30
  2.5. Extending the DSM Method .................................................................................. 31

Chapter 3. Sequential Iteration Model ......................................................................... 33
  3.1. Sequential Iteration Design Matrices .................................................................... 33
  3.2. Finding the Expected Length ............................................................................... 37
  3.3. The Minimum Length Ordering ............................................................................ 42
    3.3.1. Orderings for the 2x2 System ....................................................................... 42
  3.3.2. Variance of the 2x2 System Length ................................................................ 44
  3.3.3. Finding the Optimal Ordering for Larger Systems .......................................... 46
    3.3.4. Approximate Solutions Obtained by Heuristic Search ................................. 50
  3.4. Discussion ............................................................................................................ 53
  3.5. Conclusion ............................................................................................................ 56

Chapter 4. Work Transformation Model ...................................................................... 57
  4.1. Introduction to the Work Transformation Matrix ................................................ 57
  4.2. Work Transformation Model Development .......................................................... 58
    4.2.1. Work Transformation Model Assumptions ................................................... 59
    4.2.2. Eigenvalue Decomposition ......................................................................... 61
    4.2.3. Interpreting the Eigenstructure .................................................................... 62
      4.2.3.1. Real Eigenvalues .................................................................................... 64
      4.2.3.2. Complex Eigenvalues .......................................................................... 65
      4.2.3.3. The Eigenvectors .................................................................................. 67
Chapter 1. Introduction

1.1. Overview of Engineering Design

Engineering design is the specification of a technological product or process to meet a perceived need. Design, so defined, occurs in all engineering disciplines and industries. The success or failure of product and process designs are an important determinant of competitive success for manufacturing firms.

Research in design is conducted in many fields, and design research is defined differently in each. Among these are: psychology, where design is seen as a process of individual creativity [Clement 1989]; computer-aided design, where researchers are developing computer tools which assist the designer in developing and analyzing the product being designed [Cutkoski et al. 1989]; optimization, where design is seen as the satisfying of a goal or set of goals subject to mathematical constraints [Bell and Taylor 1992]; and management, where design is viewed as a process involving the control of an organization which is responsible for producing the design.

Although the importance of the other views of design is recognized, this thesis is an attempt to develop the management view of engineering design. We are concerned with the coordination of the many interrelated problem-solving tasks involved in a large design project. Our research approach is to use management science research methods to begin to approach this problem.

The time that it takes to design an artifact is an important determinant of commercial success [Clark and Fujimoto 1991]. Products which are designed more quickly are better able to incorporate the latest technologies and features, as well as the current knowledge about the
market. Yet it is not easy to speed the process because engineering design involves complex interactions of multiple areas of technical knowledge along with external concerns, such as consumer preferences and regulatory issues. For example, a new commercial airliner might incorporate new materials, more fuel efficient engines, sophisticated control systems, and reduced engine exhaust emissions. Combining these areas of new technology into an integrated product is difficult. Since there are potentially many interrelationships among the technical issues, it is not possible to look at each of these technical issues in isolation. Also, the interrelationships affect the amount of time that it takes to integrate the issues into a complete design.

Much recent research work on development time focuses on concurrent engineering [Nevins and Whitney 1989]. This is a philosophy in design that suggests working on more design issues simultaneously will lessen the amount of time to complete the development process. The concurrent engineering process is difficult to coordinate when the technology is highly complex.

Time is easily quantified and measured, which is an advantage when considering the application of mathematical techniques. This thesis concentrates on time-based models of engineering design which explicitly consider the effects of technical complexity on the time it takes to complete the design process.

1.2. The Design Process

The design process is the process by which a technical solution is found to meet the criteria of a given problem. For any reasonably complex problem, the design process includes the coordination of many different
tasks. The way that a design project is divided into tasks and that the tasks are related is known as the structure of the problem. How a problem is structured is an important variable in management of the design process [von Hippel 1990].

One school of thought on design defines a general structure which describes the proper design process for many engineering problems [Pahl and Beitz 1988, Hubka 1980].

Pahl and Beitz define four stages of a design process. These are, in order: product planning, conceptual design, embodiment design, and detail design. Each stage is commenced only when the previous stage is completed. Theirs is a highly sequential, strictly ordered view of design. They see design as a process in which imposed order enables the production of a superior design result. The order enables the designers to concentrate on appropriate design issues at the appropriate stages in the process and minimizes the chance that significant design issues will be overlooked.

Hubka's philosophy is similar to Pahl and Beitz, if somewhat different in the details of the model of engineering design he describes. There are instead six main stages to the generic design process: problem elaboration, establishment of functional structure, establishment of concepts, preliminary layout, dimensional layout and detailing. His rationalization of the design process is intended to make design decisions concrete and well-founded, so that better design decisions can be made and better designs produced.

Both of these models emphasize progressing through a sequence of steps in order to produce the final design. In practice, the progress on a design project is not unidirectional. We have observed that iteration occurs
on all design projects. Models of the design process concentrate on the sequential nature of the various stages through which a design process. Alternatively, the models in this thesis focus on iteration as an important activity within the design process.

1.3. Iteration in Engineering Design

Iteration in engineering design has been defined as an operation where "in a procedure similar to that used in mathematics for the approximate solution of a system of equations, one makes certain assumptions in order to proceed towards (e.g. calculate) a solution. In the next stage, these results when used as improved assumptions help to determine a more accurate solution (values), and if convergence is sufficiently rapid, a few cycles (iterations) deliver the appropriate solution" [Hubka 1980, p. 34]. Iteration is typical of complex design projects, because a one-pass solution procedure is impossible.

Although Pahl and Beitz attempt to describe design as a sequence of distinct phases, they do recognize that there is a need for iteration, particularly within each design phase. "At every step, a decision has to be made as to whether the next step can be taken or whether previous steps have to be repeated. Continuing right to the end only to discover that a serious mistake has been made at an earlier stage is something that must be avoided at all costs" [Pahl and Beitz 1988, p. 40]. They do not give any concrete description of this repetition process.

Iteration has been more explicitly considered in software design than in the design of mechanical components [Wileden 1986, Tully 1986, Curtis 1986]. This increased emphasis might be due to the important role that iteration plays in the finished product, and the degree that software
engineers use the idea of iteration to understand their environment. In any event, both the causes of and the implications of iteration in software design and design of mechanical products are fundamentally similar.

The literature discussing iteration in design suggests that there are two reasons why iteration is necessary [Tully 1986]. First is that there are many interrelated design issues, and if they were to be resolved sequentially (assuming a non-iterative sequential process were feasible) it would take more time than a simultaneous approach, even if the simultaneous approach induced some iteration. The second is that some iteration is inherently unpredictable, since some results, such as a prototype failing a feasibility test, are unforeseen and will require the repetition of previously completed design tasks (this form of iteration is also known as backtracking.)

1.4. Existing Models for Analyzing the Design Process

It is the goal of this thesis to develop techniques for modeling engineering design iteration. There are two types of model with which we are concerned. First is the information processing model of design; second is a time-based model of project management.

An information flow model of engineering design is one in which looks at design as a number of discrete information processing tasks. Each of the tasks is an information processing step. Each task requires information, produces information, and transforms information during its portion of the design process. Information processing models are used by researchers to understand the design process, but are not typically used in practice.
Among information flow models there are two strategies: dividing up the information hierarchically, and leaving it on one level. Hierarchical models can show much detail about individual events. Since there are relatively few items on any given level of the hierarchy, this structure enables much detail to be shown at any level. The standard hierarchical model of information flow is Structured Analysis [Ross 1977]. The majority of the information flow models of design are flat, as are the models presented in this thesis.

Time-based models of design are primarily standard project management tools. These are seen as somewhat limited although useful models.

These two modeling philosophies encompass distinct sets of previous models. The next two sections discuss previous models of the two types.

1.4.1. Information Flow Models

This section discusses how non-hierarchical, flat models of information flow in engineering design can show overall structure of the design problem. The advantage of flat models of information flow is that the entirety of the design structure can be seen in one view. Details about the lower levels of structure are not hidden.

An important starting point in considering the information structure in design is Alexander [1964]. An architect by training, Alexander brings a philosophical overview of how the structure of form and function are related in design solutions to human problems. Alexander advocates that there should be a large analytical component of design. He claims design analysis was not necessary in earlier phases of history, when human needs and designs were changing slowly. Current needs and technologies are changing sufficiently rapidly such that analysis is beneficial.
Figure 1. Linear Graph with Decomposition

Alexander's treatise is divided in two sections, the first of which describes the reasoning behind trying to build a descriptive framework, and the second of which is the framework laid out formally. An appendix contains a fully worked example of applying the framework to designing a village which meets the needs of its residents.

Figure 1 shows the type of model that he develops of design. Each of the nodes in the graph is a technical issue, and each arc represents an instance where two issues affect each other either in mutual support or in conflict. The graph is decomposed by a mathematical technique into subsections which have relatively few interactions between the subsections. The decomposition suggests design issues which should be addressed by the same or similar part of the solution space.

Alexander's model is most useful in identifying subsystems within a large and complex project. He argues that all diagrams or models are abstractions, but the abstractions give us useful working knowledge about
the problem we are trying to solve. There is a hierarchy to the levels of the problem that is to be solved, and there is a hierarchy to the technical solution realized. If these hierarchies are matched, then the solution will best fulfill its goals, minimizing the incongruities between context and form.

Lewis et al. [1973] incorporate Alexander's type of analysis into a computer package which will suggest groupings of tasks based on their known relationships. The paper does not add any theoretical modeling to Alexander's technique, but accepts the technique as given. The paper describes an application of the method to the design of a building housing laboratories and offices. They suggest that the method can identify 'good' groupings of design parameters which are not perceived by an expert designer, where a good grouping indicates the direction where a designer's effort may be most profitably employed.

Lewis suggests that this type of analysis is most beneficially applied to problems which are large enough so that the relationship structure is not obvious to an expert, but the relationships are well enough understood that the analysis has some correspondence with the actual problem. This would be the case when a designer wishes to gain a fresh look at a familiar problem, or when a new problem is sufficiently akin to a known problem such that the relationships may be identified beforehand.

Steward [1981] extends the structural analysis of design using formal mathematical and graph theoretical techniques. His modeling technique, known as the Design Structure Method (DSM) serves as the basis for the models developed within this thesis. This method is discussed more fully in Chapter 2. Steward's work arises from previous work in solving large systems of equations. He noted that setting design parameters in a complex
design project was similar to solving for individual variables in a system of equations. Some design parameters can be set independently, and some need to be solved for simultaneously. When applied to a design problem the relationships can be represented as a directed graph. The type of graph in Steward's representation differs from Alexander's in that the arcs are directed, where an arc represents an information flow between design tasks (nodes).

The information in the graph can be represented more compactly in a matrix. Manipulation of the rows and columns of the matrix can identify which are the design parameters which can be solved independently, and which need to be found concurrently. Steward also describes a solution strategy for solving the concurrent variables based on the structure of the graph.

Steward presents an interesting technique, but it is weak in application to real problems, or in understanding the underlying organizational and engineering aspects of finding a design solution. The applications within his book are not of full industrial problems. The research group of which I am a part has found several questions which arise in application which are not explained in the book [Gebala and Eppinger 1991, Eppinger et al. 1992, Eppinger 1991]. These issues include the distinction between tasks and parameters in the matrices, and quality of matrix data.

Marple [1961] describes the problem solving process undergone during a typical design project. The paper is divided in two sections: the first where the decision making process is documented for several design projects, and the second where the decision making process is analyzed and a framework is given which describes the process. The design projects
are related to gas cooled nuclear reactor design. One of the conclusions of Marples' paper is that the engineer searches for innovations in a rational manner. It is possible to see its link with, and refutation of, early works on innovation, where the 'great men' theories were in vogue. Also, the point is made that designers are not required to be 'creative' unless all known, tried-and-true solutions have been shown to be insufficient.

In the second part of the paper, the decision making process is written down in a formal way. It is suggested that the structure of these design processes is typical of engineering design. The search procedure occurs in a tree structure, where each technical solution to a problem gives rise to its own sub-problems. (See Figure 2.) Initially, there is the original design goal and several alternative designs. Each of the alternatives gives rise to one limb of the tree. Every alternative has its own set of technical hurdles to overcome, which also lead to their own sub-branches. Each branch of the tree must be evaluated until a technically acceptable solution is found.
Figure 2. Decision Tree

It is not possible to construct such a decision tree prior to the design phase. Instead, the tree is used to give order to a completed decision
making process which seems unstructured. Since every tree is unique, the applicability of this type of model is limited as a tool for project management, since every manifestation is not predictable in advance.

Ramström and Rhenman [1965] present a different kind of model than Marples, but one which is used to the same effect. They document the decisions made during a typical engineering design project, and then attach a descriptive framework. Instead of having a tree which represents progress, each decision is seen as a transformation in the range of technical alternatives available.

The solution to every engineering design problem is manifested in a number of technical dimensions, such as its material, its geometry, its control system, and its cost. The solution space initially incorporates a range of possibilities on each of these technical dimensions. Every decision taken during the design process transforms the solution space. These transformations may be limitations (restriction of the set of possible alternatives), generalizations (opening up to new possibilities), or changes (discarding of some alternatives while adding others.) Eventually, the range of the technical dimensions will be reduced to a single alternative, and the design will have been completed.

The Ramström and Rhenman model is also applied to a small design problem within a nuclear reactor project. It is an example of a design project which is neither routine (solution known in advance) nor unusual (requires a solution outside of the engineers' expertise). It illustrates that the range of technical alternatives is in flux during the design phase. The way that the changes occur cannot be predicted in advance, but it can be predicted that there will be changes.
The Luckman [1967] model of engineering design is a formal and analyzerable model, but a model which is not closely tied to actual problem solving. (The model was also presented briefly in [Harary et al. 1965].) The Luckman model is a graph theoretic model, where relationships between components are represented by arcs between nodes. (See Figure 3.) Each node contains several technical alternatives (such as whether to have load bearing or non-load bearing external walls), which may be compatible or incompatible with technical alternatives at other nodes. Every arc represents a pair of alternatives which are technologically incompatible. The job of the designer is to identify a set of alternatives which are fully compatible. Presumably, it would be possible to search through the various alternatives in some efficient way.

Figure 3. Option Graph
The Luckman model is a normative model, but it has little applicability. It is too difficult to determine what all the alternatives are beforehand, and it is equally difficult to know what the relationships are between those alternatives. If the problem were well enough known to build such a graph, the problem would already be solved. It may be possible to extend Luckman's model to make it more representative of actual problems, but I am unaware of any extensions in the literature.

The information flow models are intended to recognize the structure of the design problem, and to suggest how the structure of the problem can be controlled and improved. These models do not, however, treat time in an explicit manner. Models which do treat time explicitly are discussed in the next section.

1.4.2. Time-Based Models

There is an alternative set of models which are used to estimate time of design projects. These models are the standard tools of project management. These models are divided into two classes, acyclic and cyclic.

The standard project management models, CPM and PERT (which are substantially similar) are acyclic in that the structure of the flow among the tasks is unidirectional and no tasks are repeated. CPM/PERT are network models which are used to schedule the project tasks and to determine capacity and other resource constraints. A good reference to standard project management techniques is [Nicholas 1990].

These models do not work well for engineering design because the iteration inherent in engineering design creates looping within the network project models, which cannot be accounted for in the traditional techniques. GERT, a more general and powerful technique, allows such looping within
its network. Direct analysis of any but a simple GERT network rapidly becomes unwieldy, so simulation is typically used to evaluate a project. (Taylor and Moore [1980] discuss how GERT can be applied to R&D projects.) Because of the way GERT networks are analyzed, it is difficult to see how the structure of the project network affects development time.

One recent attempt exists which extends GERT models for product development [Adler et al. 1992]. This model recognizes that GERT does not account for delays which are caused by multiple product development projects' needing similar scarce resources, thereby delaying each other. The model is analyzed using simulation techniques. That work is a substantially different extension to project management tools from this thesis, but it does acknowledge that there are limitations to application of GERT to product development projects.

Dougherty et al. [1984] discusses how project management tools are used in the management of development projects. Their conclusions are that R&D project managers typically use simple models for project management, despite the availability of more complex models. The perception is that the amount of work that it takes to use a more sophisticated model overwhelms the benefit from its added power.

1.5. What this Thesis Does

The goal of this thesis is to develop mathematical models of design process which include the aspects of the information flow structure of the design problem (as described in section 1.4.1) and development time (as described in section 1.4.2). These models are developed using the idea that iteration is a fundamental and ubiquitous feature in the design of complex artifacts.
The remainder of this thesis describes the models of design iteration. Chapter 2 gives greater background on the Design Structure Matrix method, which serves as a basis for the iteration models. Chapter 3 describes the Sequential Probabilistic model of iteration, and shows how the order of the tasks within an iterative block affects completion time. Chapter 4 presents the Work Transformation Matrix model and its applications. This model introduces the idea of the design mode, a group of tasks which jointly consume the majority of the time during an iterative design process. The application of Work Transformation analysis to brake system design and electronics module design describes how the design modes are identified and how these modes are integral to the amount of time that it takes to complete the design process. Chapter 5 discusses how it is possible to know whether the models within this thesis are a valid description of the design iteration process. Chapter 6 contains the conclusion and indicates further areas of work which are suggested by the thesis research.
Chapter 2. The Design Structure Method

The goal of this chapter is to introduce the concepts which are the basis for the Design Structure Matrix (DSM) method and to describe various ways the method has been applied. This technique was described briefly in Chapter 1 as a static model of the design process which can identify precedences and interrelationships within a complex engineering design project. The DSM method serves as the basis for the theoretical extensions of iteration models described in Chapters 3 and 4.

2.1. The Design Structure Method

The assumption which underlies the Design Structure Matrix method is that the information flows within an engineering project are known and these flows give some structure to the entire project. In order for this assumption to be valid, there are a number of implications. First, the project must be divisible into discrete, known tasks. The informational relationships (who needs information from whom) must also be known. In order for the tasks to be known, the technology must be well understood by the people responsible for managing and executing the design project.

Let us consider the information relationships between two design tasks. There are three possibilities. First, that there is a unidirectional information need, where one task needs information from the other, but not the converse (Figure 1a). Second, that there are no informational needs between the tasks, they are independent (Figure 1b). Third, that there is a circular need for information, the each task needs information from the other (Figure 1c).
(a) Dependent    (b) Independent    (c) Interdependent

Figure 1. Possible Information Flows for Two Tasks

Tracing these information flows for larger graphs becomes rapidly unwieldy. The same information can be written more compactly and understandably in matrix form. Each row and corresponding column in a square matrix is identified with one design task. Each off-diagonal element is an information dependency (similar to the directed arcs in Figure 1.) The diagonal elements do not convey meaning, they are place holders so that the diagonal can be identified easily. This matrix is known as the Design Structure Matrix.

The matrix equivalents to the three graphs shown in Figure 1 are given in Figure 2.

(a) Dependent    (b) Independent    (c) Interdependent

Figure 2. DSM Equivalents for Two Task Information Flows

It is easy to recognize from the matrices where the information dependencies are, and the relationships between the tasks.

As an example Design Structure Matrix which is somewhat larger, consider the camera design matrix shown in Figure 3. It is possible to
identify within the matrix which tasks fit into which of the three categories described above.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Set Specifications</td>
<td>Design Concept</td>
<td>Design Shutter Mechanism</td>
<td>Design Viewfinder</td>
<td>Design Camera Body</td>
<td>Design Film Mechanism</td>
<td>Design Lens Optics</td>
<td>Design Lens Housing</td>
</tr>
</tbody>
</table>

![Matrix Diagram]

**Figure 3. Camera Design Matrix**

There are eight tasks in the camera design problem. The middle four tasks (C-F) are coupled. Since task A must precede task B, A and B are sequential. The four coupled tasks (C-F) can be done in parallel with the sequential pair of tasks G and H.

The directed graph description and its matrix equivalent are the basis for the Design Structure Matrix method. The applications considered are in engineering design of manufactured products. Similar analytical techniques have been used in chemical engineering [Kehat and Shacham 1973] and computer science [Warfield 1973] to analyze the decision making process. A systems-level description of the decisions is common to all fields in which these methods have been applied, and many of the ideas developed in this thesis can be applied to other fields.

2.2. **Partitioning Design Structure Matrices**

In order to determine whether the information flows in any given project are dependent (series), independent (parallel) or interdependent (coupled) it is typically necessary to reorder the tasks in the matrix so that
the off-diagonal entries are as sub-diagonal as possible. This process is known as partitioning, since the matrix is partitioned into distinct coupled blocks, where possible.

An earlier thesis described two primary methods by which a matrix can be partitioned [Gebala 1991]. The more straightforward method is to trace the loops in the matrix directly. It is also possible to use the notion of the powers of the adjacency matrix. These two methods are described in the following paragraphs.

Tracing the loops directly involves application of the following algorithm.

**The Loop Tracing Algorithm**

The algorithm used to find the coupled block is as follows:

1. Identify any tasks which can be executed first (tasks which have no predecessors.) Eliminate them and their dependencies from further consideration. Repeat until no more such tasks found. If there are no tasks remaining, then stop.

2. Identify any tasks which can be executed last (tasks which have no successors.) Eliminate them and their dependencies from further consideration. Repeat until no more such tasks found.

3. If there are no tasks without inputs or outputs then there must be groups of coupled tasks. Start at any task and save the name of this task in a string. One of the output tasks from that task is chosen. This task is added to the string of the tasks in the current search. If this task already appears in the string, then a loop of coupled tasks has been found. If this task is not yet in the string, then continue adding tasks in this manner until a loop is found. All tasks which
appear in the string between the first and the second appearance of the repeated task are within the same coupled block. Combine all of the tasks in the coupled block into one pseudo-task. The pseudo-task has all of the inputs and outputs of any task within the pseudo-task. Go back to 1.

The other alternative is to use idea of the powers of the adjacency matrix. To use this method, the design structure matrix is transformed into a mathematical matrix \( A \) with all entries are either 1 (dependency) or 0 (no dependency). All diagonal entries are 0.

For every power \( A^n \), if entry \( a_{ii} > 0 \) then task \( i \) is in a circuit of length \( n \). Each of the successive powers of \( A \) is calculated, and it is determined which tasks are within which loops.

Either of these partitioning methods is easily implemented. After the coupled blocks are identified, it then remains to understand and manage the tasks within the coupled blocks.

2.3. Other Theoretical Issues

This section describes existing techniques for analyzing coupled blocks within a DSM. The first method, shunt diagrams, is suggested by Steward [1981] as a graphical technique for determining which dependencies' removal will eliminate the feedbacks.\(^1\) The second method is to attempt to remove the coupling by redefining the technical tasks.

\(^1\)The term 'tearing' has been used for this process. This term is not used in this thesis, because the term suggests that the dependencies are being broken. The dependencies are not being broken, rather an efficient set is being found which will lessen the quantity of feedback dependencies which are present in the matrix.
2.3.1. Shunt Diagrams

A shunt diagram is the directed graph which corresponds to the Design Structure Matrix for a coupled block. An example shunt diagram and its corresponding DSM is given in Figure 4.

![Diagram showing a shunt diagram and its corresponding DSM.]

**Figure 4. A DSM with its Corresponding Shunt Diagram**

The shunt diagram is used to identify arcs which are high leverage points; their removal will eliminate much of the cycling within the coupled block.

Steward suggests four criteria for choosing dependencies to be eliminated. These are:

1. Choose a minimum number of dependencies to be eliminated.
2. Confine the dependencies to those which can be close to the diagonal.
3. Choose dependencies in a minimum number of columns.
4. Choose dependencies in a minimum number of rows.

Finding a set of dependencies which satisfies the above criteria is computationally difficult, although some heuristics have been suggested [Steward 1981].
The shunt diagram technique is useful for specifying an ordering of coupled tasks which lessens the quantity of feedback within the matrix. The technique is somewhat weaker in incorporating the technological issues which can dominate the process of identifying why one design process might be superior to another. The technique described in the next section, as well as the techniques developed in chapters 3 and 4, incorporate greater information about the technical issues implicit in coupled blocks.

2.3.2. Redefining Tasks

Another option when attempting to solve a coupled problem is to redefine the tasks such that the coupling is removed [Krishnan et al. 1991].

\[\begin{array}{ccc}
A & B & C \\
A & \times & \times \\
B & \times & \times \\
C & \times & \times \\
\end{array}\]

**Figure 5. Coupled Design Matrix**

Redefinition of tasks will affect the information which is contained in the Design Structure Matrix. If one task is created to define the interface between two coupled tasks, it may be possible to execute them in parallel because of the results of the newly created task. Resolving coupled tasks through redefinition is a useful design strategy since it simplifies each of the design decisions to be made [Suh 1990]. Figure 6 displays how the matrix in Figure 5 might appear after creating a new task, Z, which defines the interface between A and C to remove coupling.
Whether or not such redefinition is possible or desirable is a difficult question, in general. The source of the redefinition is specific to the domain in which it is applied.

2.4. Applications of the DSM Method

This section looks at other attempts in the literature to apply and extend the Design Structure Matrix method. These efforts suggest that the DSM method is a useful tool to document and examine the structure of a design project. They also suggest that the method needs to be extended if it is to be useful to managers to improve their ability to use the method to control a design project.

2.4.1. Black

Black [1990] applied the Design Structure Matrix method to automotive brake system design. This detailed analysis of low level design decisions identified one hundred parameters which are chosen during the design process. Thirty-four of these parameters are in one large coupled block which is at the heart of the design problem. (The brake system design problem serves as an example for the design iteration model discussed in Chapter 4.)

Black suggests that the Design Structure Matrix provides a useful codification of systems design knowledge, and a good tool for developing a
consensus within the design organization on what the interrelationships between components are.

2.4.2. NASA

Several researchers at NASA have been using the Design Structure Matrix to analyze design processes with the goal of design process improvement [Rogers 1989, Sobieszczanski-Sobieski 1989]. Rogers has been concentrating on developing an expert system which can perform the partitioning and shunt diagram analysis described in Sections 2.2 and 2.3; Sobieszczanski-Sobieski has combined the Design Structure Matrix framework with a mathematical description about the physical problem to form a hierarchical, constrained nonlinear optimization problem which can then be solved to find an optimal design.

2.4.3. Sequeira

Sequeira [1991] uses Design Structure Matrices to analyze the early phases of the automobile development process. She suggests that the DSM is a useful tool for identifying and controlling the information flows in engineering design.

In the automobile development process she identifies four primary groups of coupled tasks, Concept Development, Theme Development, Part Development and Tooling Development. In the suggested improved design process, more design analysis is undertaken early in the design process, with a reduction in the amount of feedback later in the design process. It should be noted that the improvements to the design process require changes to the tasks, and are not merely changes in how the current tasks are organized.
2.5. Extending the DSM Method

This thesis provides two new extensions to the Design Structure Matrix Method. They are both attempts to use the DSM to estimate the time that it takes to complete a coupled block. There are several ways that an coupled block can be interpreted. Included among these are iteration, negotiation, simultaneous solution, and removing the coupling through task redefinition.

The extensions in this thesis are based on the idea that a coupled block describes an iterative solution to solve the technical problem. Negotiation can be viewed as a specific type of iteration where the participants are successively proposing solutions for the technical problem, and each proposal is a new iteration. Simultaneous solution would require that all of the tasks are sufficiently well understood such that they can be reduced to one set of tractable constraints, which is typically not the case. For many problems it is not possible to redefine the tasks (as suggested in section 2.3.2) to eliminate the coupling. Iteration is therefore necessary, since it is not possible to solve the problem in a once-through procedure.

Models which predict iteration time are not well developed. It must be noted that the iteration precludes the use of standard project management tools such as those described in section 1.4.2. Series tasks can be evaluated by summing their individual times, and parallel tasks can be evaluated by finding the maximum of those task times, but the coupled tasks require a new method. For the project characterized by the DSM in Figure 3, if the task time are \( a, b, c, \ldots, h \), the time of the camera design project would be:
a + b + \max\{ f(c,d,e,f), g+h \}

where \( f(\cdot) \) is a function, undefined as yet, corresponding to the development time for the coupled block.

The models presented in this thesis suggest ways that iteration time can be evaluated for such a coupled block of tasks. The model in chapter 3 is a probabilistic model, where the number of iteration cycles is a random variable. The model in chapter 4 is a deterministic rework model, where every design decision is repeated many times, but the amount of work done during each iteration is decreasing.
Chapter 3. The Sequential Probabilistic Model

This chapter develops a model which attempts to describe the iterative solution progress experienced in complex engineering design projects. The task model is based on the Design Structure Matrix and assumes that each separable design task takes a finite, known amount of time, but must be repeated a number of times in order to resolve precedence conflicts. This type of design process is called sequential iteration. The probability of repetition is a function of how tightly coupled the tasks are. Based on the model, a timing score is computed, and it is possible to identify a suggested initial ordering of the design tasks which minimizes the expected length of time that it takes to resolve the coupled design problem. Limitations and extensions to the model are also discussed.

3.1. Sequential Iteration Design Matrices

The model developed in this chapter allows us to organize a group of coupled design tasks so that the time that it takes to complete the design process is lessened or minimized.

In order to estimate the time necessary to complete a coupled block, it is necessary to assign a time to each of the individual tasks, and also to develop a scheme which shows how the tasks are interrelated. The model in this chapter is a sequential iteration model, which assumes that the tasks in a coupled block are done one at a time (sequentially), and there is a finite probability that coupled tasks will have to be repeated (iteration).

The original Steward Design Structure Matrix is replaced by a quantitative DSM. The diagonal elements of the quantitative DSM indicate the length of time that the task requires. This time is independent of the
number of times that a task has been repeated, and of the amount of information available. It is assumed that the task duration is known *a priori* and is deterministic. In place of the ordinary DSM's binary elements, the off-diagonal terms now correspond to probabilities, and are therefore on \([0,1]\). Each off-diagonal value \(p_{ij}\) indicates the probability that another iteration of task \(i\) will be necessary given that task \(i\) was performed without the knowledge of the latest results from task \(j\).

As an example consider the matrix in Figure 1.

![Matrix Diagram](image)

**Figure 1. Sequential Iteration Design Structure Matrix**

Task \(A\) takes 4 units of time if done in isolation. Task \(B\) takes 7 units if done in isolation. Tasks \(A\) and \(B\) are coupled such that if \(A\) is done before \(B\), then there is a probability of 0.2 that \(A\) will have to be repeated (taking another 4 time units) because the results of \(B\) are incompatible with the previous results of \(A\). If \(B\) is initially done before \(A\), then there is a 0.4 probability that \(B\) will have to be repeated later. Furthermore, each time one of the tasks is performed, the other task may need to be repeated also, based on these probabilities. The total number of iterations is a random variable.

It is assumed that only one of the tasks is performed at a time, so this is a purely sequential design procedure, although repetition may be necessary. (The term sequential iteration is used to distinguish this from a purely sequential once-through procedure.) Because all work is sequential, the total duration of the coupled design tasks is equal to the total amount of engineering effort (in person-time units) that is spent on those tasks.
It is also assumed that there is no delay between the task executions. This implies that there are no resource constraints, and that information flows are instantaneous. Situations which might cause this assumption to be violated include executing several projects simultaneously, such that each task (or server) must choose which of the projects to work on. Alternatively, there might be some other constraint on why a resource is unavailable, such as task downtime, or purchasing lead time.

The decision maker (engineering manager) chooses the order that tasks are attempted. The choice of order affects the distribution of the completion time. For an nxn matrix, there are n! candidate orderings. The manager must choose the ordering which has a desirable distribution of the completion time. The expected length of the completion time can be readily calculated, and serves as the primary criterion for evaluating orderings.

Under this interpretation, the matrix can describe a decision Markov chain, where the decision maker can choose the ordering of the initial attempt at completing the coupled tasks. Each state in the chain corresponds to completing one task at one time, and the transition probabilities correspond to the probabilities of repeating a previous task, or of attempting a new task. There is not a one-to-one correspondence between tasks and states. The expected length of the Markov chain can be evaluated and used to determine which of the orderings is preferable. The Markov chain in Figure 2 corresponds to the quantitative matrix in Figure 6. The ordering decision is equivalent to specifying which of the two dashed arcs to include in the chain if the total expected length is to be minimized.
Figure 2. Decision Markov Chain for Sample Matrix

In order to preserve the Markov chain description for larger groups of coupled tasks, it must be assumed that if the results of a downstream task are incompatible with previous results, it is possible to resolve the conflict between the most recently completed task and the previous tasks by repeating only one of the previous tasks. When the Markov chain interpretation is preserved the mathematics are well understood and tractable. An example 3x3 matrix is given in Figure 3. (Maintaining the Design Structure Matrix description, a blank entry corresponds to a null repeat probability.)

Figure 3. 3x3 Numerical Design Structure Matrix

Upon completion of a task, a probabilistic choice is made of all previous tasks in the matrix. Either one of them is chosen for iteration, or else the task is a success and the subsequent task is attempted. In order for a Markov chain interpretation to be valid the probabilistic (off-diagonal) elements in any one column can add up to no more than one.
3.2. Finding the Expected Length

It now remains to calculate the expected length of any candidate ordering. The following discussion illustrates how the length may be efficiently calculated using a modified form of Gaussian elimination. As an example, consider the 3x3 matrix from Figure 3, and we will calculate the length of the ordering A-B-C. The Markov chain for that ordering is shown in Figure 4.

![Markov Chain Diagram]

Figure 4. Markov Chain for 3x3 Sample Matrix

The chain is divided into the three stages which correspond to the cascading process of task completion. Task A is executed first, with a probability of 1.0 of successful completion (at the current stage.) Task B is then attempted, which succeeds with a probability of 0.8. Task A must be repeated with a probability of 0.2. A and B then iterate until their results are compatible, at which time task C is executed for the first time. After task C, there is a probability of 0.5 that the design is complete, but with a probability of 0.5 there must be iteration among A, B and C to complete the project.

To find the expected length of time that it will take to complete the process, the computation is started in the third stage of the chain. It is possible to compute the expected time remaining for each of the nodes in the
third stage. Since it is known that the third stage starts at node C, we only need to compute the expected length of time remaining from this starting node.

Similarly, the expected length of time that the process will spend in the second stage of the chain is then calculated. We must determine the expected time the process will spend in the second stage given that this stage starts in node B of stage two. The value for stage two is then added to our earlier value of node C from stage three. The length of task A is then added, since there is no iteration in stage one.

Algebraically, the calculation is similar to Gaussian elimination (since the expected times can be written as a set of linear equations), although standard Gaussian back substitution is not used. (For details on calculations with Markov chains, see Ross [1983].) There are \((n^2 + n)/2\) states in the Markov chain corresponding to a \(n \times n\) matrix. Standard Markov chain analysis would therefore require solving \((n^2 + n)/2\) simultaneous linear equations. The calculation procedure presented below (which uses non-standard back substitution) is only as complex as solving \(n\) simultaneous linear equations.

Let \(r_A\), \(r_B\) and \(r_C\) be the total expected time remaining at each of the three nodes in the third stage of the chain. Those three quantities are related by:

\[
\begin{align*}
  r_A &= 0.4 \, r_B + 0.3 \, r_C + 4 \\
  r_B &= 0.2 \, r_A + 0.1 \, r_C + 7 \\
  r_C &= 0.5 \, r_B + 6
\end{align*}
\]

which can be rewritten as
\[
\begin{bmatrix}
1 & -0.4 & -0.3 \\
-0.2 & 1 & -0.1 \\
0 & -0.5 & 1
\end{bmatrix}
\begin{bmatrix}
r_A \\
r_B \\
r_C
\end{bmatrix}
= \begin{bmatrix}
4 \\
7 \\
6
\end{bmatrix}
\]

Solving by Gaussian elimination obtains

\[
\begin{bmatrix}
1 & -0.4 & -0.3 \\
0 & 0.92 & -0.16 \\
0 & 0 & 0.91
\end{bmatrix}
\begin{bmatrix}
r_A \\
r_B \\
r_C
\end{bmatrix}
= \begin{bmatrix}
4 \\
7.8 \\
10.23
\end{bmatrix}
\] (*)

Since iteration begins at node C during the third stage we need to retain the value for \( r_C \).

\[ r_C = \frac{10.23}{0.91} = 11.21 \]

We do not need to retain the values for \( r_A \) and \( r_B \) since it is known that the third stage will be entered at node C.

Similarly for the two nodes in the second stage of the Markov chain, where \( s_A \) and \( s_B \) are the expected times spent in the second stage at each of the two nodes,

\[
\begin{bmatrix}
1 & -0.4 \\
-0.2 & 1
\end{bmatrix}
\begin{bmatrix}
s_A \\
s_B
\end{bmatrix}
= \begin{bmatrix}
4 \\
7
\end{bmatrix}
\]

Solving by Gaussian elimination we obtain

\[
\begin{bmatrix}
1 & -0.4 \\
0 & 0.92
\end{bmatrix}
\begin{bmatrix}
s_A \\
s_B
\end{bmatrix}
= \begin{bmatrix}
4 \\
7.8
\end{bmatrix}
\]

Since iteration begins at node B during the second stage we need to retain the value for \( s_B \)

\[ s_B = \frac{7.8}{0.92} = 8.48 \]

And likewise, in the first stage of the Markov chain, where there is only one node, with task duration \( t_A \),

\[ t_A = 4 \]

The total expected time for the Markov chain defined in Figure 4 is 23.69. (Sum of \( r_C \), \( s_B \) and \( t_A \)). It is observed that the equations for \( r_C \), \( s_B \) and
$t_A$ are expressed in the Gaussian elimination of the equations in the third stage (Equation (*) above.) The expected times $t_C$, $s_B$ and $t_A$ are the right hand side values divided by the diagonal of the matrix on the left (ignoring the entries of the matrix above the diagonal.) This property is true in general because the earlier stages of iteration are sub-problems of the final stage. This special structure is exploited to establish the efficient calculation algorithm for the expected time through the entire network.

**An Efficient Length Computation Algorithm**

**Step 1: Transform the Design Structure Matrix into Markov Chain form.**

In place of using the quantities in the Design Structure Matrix directly, we must transform it into a matrix $P$ which describes the Markov chain. This is done by taking the negative of the transpose of the DSM, and inserting ones along the diagonal.

$$p_{ij} = \begin{cases} -dsm_{ji} & i \neq j \\ 1 & i=j \end{cases}$$

**Step 2: Place task times in a column vector.**

The diagonal elements of the DSM, which contain the times of each of the tasks, are placed in a separate vector $b$.

$$b_i = dsm_{ii}$$

**Step 3: Diagonalize matrix $P$.**

Ordinarily, the equation $Px=b$ is subjected to Gaussian elimination in order to solve for $x$. Gaussian elimination finds the unique $L$, $D$ and $U$ (lower triangular, diagonal, and upper triangular matrices, respectively) such that $LDU=P$. 

40
Step 4: Use modified back substitution to find $x'$.

$b'$ is calculated where $b' = L^{-1}b$. Normal back substitution solves (trivially) $DUx = b'$ for $x$. Each of the elements in $x$ contains the remaining value at each of the nodes in the final stage of the Markov chain. Since we only need to know the remaining time at the starting node of the final stage, and we also need the times at the starting node of each of the other stages, we can accomplish this by instead solving $Dx' = b'$ (which is also trivial since $D$ is diagonal.)

$$x' = D^{-1}L^{-1}b$$

Step 5: Add elements in $x'$ to get total expected time.

Each of the elements of $x'$ contains the expected length of time to be spent in each of the successive stages of the Markov chain, given that the entering node is the node not seen in any of the previous stages. Therefore the expected length of time that the current ordering will require is the sum of the elements in $x'$.

$$\text{Length} = \sum_{i=1}^{n} x'_i$$

Since each of the elements in $x'$ is at least as large as the value of the entering task by itself ($b_i$), the sum of the task times is a lower bound on the value of any ordering. The lower bound would appear in practice if all of the probabilistic choices between success or repetition of a previous task were successful.

If the Markov chain has only transient states (there exists a positive probability path from every state in the Markov chain to the finish state), then the expected length of the design process is finite. This will be the case
if the matrix $P$ is positive definite (all the diagonal elements of $D$ are positive.) The solution process will therefore calculate the expected length without finding zero pivots for a realistic process.

3.3. The Minimum Length Ordering

For the 3x3 example given above, all possible orderings and their expected times are listed below.

<table>
<thead>
<tr>
<th>Ordering</th>
<th>Expected Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-B-C</td>
<td>23.69</td>
</tr>
<tr>
<td>A-C-B</td>
<td>20.43 (minimum)</td>
</tr>
<tr>
<td>B-A-C</td>
<td>22.66</td>
</tr>
<tr>
<td>B-C-A</td>
<td>28.54 (maximum)</td>
</tr>
<tr>
<td>C-A-B</td>
<td>22.23</td>
</tr>
<tr>
<td>C-B-A</td>
<td>25.54</td>
</tr>
</tbody>
</table>

For this example, the minimum length ordering (A-C-B) is 40% shorter than the longest ordering (B-C-A). As is demonstrated in the next section, the minimum length ordering exhibits the properties of having the longer tasks later in the process, and having the larger repeat probabilities subdiagonal. (See Figure 5.)

![Figure 5. Optimal Ordering of 3x3 Example Matrix]

3.3.1. Orderings for the 2x2 System

As an example of optimizing a sequential iteration process, let us analyze a hypothetical 2x2 system with symbolic entries in the matrix (See Figure 6.)
Figure 6. Symbolic 2x2 Matrix and Markov Chain for Ordering A-B

There are two candidate orderings, A-B and B-A. By performing the calculations as indicated in the previous algorithm:

\[
P = \begin{bmatrix} 1 & -w \\ -z & 1 \end{bmatrix} \quad b = \begin{bmatrix} x \\ y \end{bmatrix}
\]

\[
L = \begin{bmatrix} 1 \\ -z & 1 \end{bmatrix} \quad D = \begin{bmatrix} 1 & -wz \\ 1-wz & 1 \end{bmatrix} \quad U = \begin{bmatrix} 1 & -w \\ 1 & 1 \end{bmatrix}
\]

\[
D^{-1}L^{-1}b = \begin{bmatrix} x \\ y + xz \\ \frac{1}{1-wz} \end{bmatrix}
\]

Under ordering A-B the expected length of time that it takes to complete the design process is

\[
x + \frac{y + xz}{1 - wz}
\]

Similarly, the expected length for ordering B-A is

\[
y + \frac{x + yw}{1 - wz}
\]

**Theorem:** Ordering A-B will have a lesser expected time than B-A if and only if

\[
\frac{z}{1 - z} x < \frac{w}{1 - w} y.
\]

**Proof:** Based on the above discussion, ordering A-B will have a lesser expected time than B-A if and only if
\[ x + \frac{y + xz}{1 - wz} < y + \frac{x + yw}{1 - wz} \]

which can be rearranged to

\[ x - xwz + y + xz < y - ywz + x + yw \]

which reduces to

\[ xz(1-w) < yw(1-z) \]

or

\[ \frac{xz}{(1-z)} < \frac{yw}{(1-w)} \]

which completes the proof.

Looking at the above theorem, it is seen that it is preferable to have the shorter tasks earlier (x smaller than y), and to have the higher repeat probabilities below the diagonal (z smaller than w). For example \( \begin{bmatrix} \text{1} \\ \text{.5} \end{bmatrix} \begin{bmatrix} \text{2} \\ \text{.5} \end{bmatrix} \) is preferred to \( \begin{bmatrix} \text{2} \\ \text{.5} \end{bmatrix} \begin{bmatrix} \text{1} \\ \text{.5} \end{bmatrix} \), and \( \begin{bmatrix} \text{1} \\ \text{.3} \end{bmatrix} \begin{bmatrix} \text{1} \\ \text{.3} \end{bmatrix} \) is preferred to \( \begin{bmatrix} \text{1} \\ \text{.5} \end{bmatrix} \begin{bmatrix} \text{1} \\ \text{.3} \end{bmatrix} \). The magnitude of the repeat probabilities has a stronger effect due to the non-linear way in which it appears in the inequality.

3.3.2. Variance of the 2x2 System Length

It is also possible to find the variance of the length of the 2x2 system. One would desire an ordering to have a low variance, so that the manager of the design process can predict with greater certainty the length of the iteration process.

The variance for the ordering A-B is calculated by looking at the length of the reward Markov chain as a sum of two independent random variables, one of which is geometric (for complete A-B iteration), and one of which is Bernoulli (for partial iterations, i.e. A only).
The geometric random variable corresponds to the number of complete A-B iterations which occur during the design process. Looking at the chain in Figure 6, every arrival to node B is considered a renewal to our geometric random variable. The probability of a renewal is wz, and the reward of each renewal is x+y. The variance of this geometric random variable is therefore

\[
(x+y)^2 \frac{wz}{(1-wz)^2}.
\]

The Bernoulli variable corresponds to the probability that node A is the final node visited given that this is the last iteration. The probability that A is the final node visited is \(z(1-w)/(1-wz)\), and the reward of that probability is x. The variance of the Bernoulli random variable is therefore

\[
x^2z \frac{(1-w)(1-z)}{(1-wz)^2}.
\]

Since the random variables are independent, the variance of the 2x2 system in Figure 6 (ordering A-B) is

\[
(x+y)^2 \frac{wz}{(1-wz)^2} + x^2z \frac{(1-w)(1-z)}{(1-wz)^2}.
\]

For ordering B-A the variance is

\[
(x+y)^2 \frac{wz}{(1-wz)^2} + y^2w \frac{(1-w)(1-z)}{(1-wz)^2}.
\]

The geometric random variable has the same distribution for both possible orderings. The distribution of the Bernoulli random variable therefore determines the difference between the variance of the two orderings.

**Theorem:** Ordering A-B will have a smaller variance than ordering B-A if and only if
$x^2 z < y^2 w$.

Proof: Based on the above discussion, it is known that ordering A-B will have a smaller variance than ordering B-A if and only if

$$(x+y)^2 \frac{wz}{(1-wz)^2} + x^2 z \frac{(1-w)(1-z)}{(1-wz)^2} < (x+y)^2 \frac{wz}{(1-wz)^2} + y^2 w \frac{(1-w)(1-z)}{(1-wz)^2}.$$ 

This expression reduces obviously to

$$x^2 z < y^2 w$$

and the theorem is proved.

Again, our process is improved if the larger diagonal values occur later in the matrix, and larger repeat probabilities are below the diagonal. The geometric random variable is, however, identical for both orderings.

The variance of a matrix larger than 2x2 is much more complicated, but we would expect that the trends remain as in the simple case. The variance will be lessened if large probabilities are sub-diagonal, and large times are late in the process, which are the same trends observed when minimizing the expected time. Therefore, lessening the variance is a general byproduct of lessening the expected time.

From a managerial perspective, lessening the variance is equivalent to lessening the risk that the project will take longer than predicted. It is therefore beneficial that the same rules of thumb which reduce expected time (large times late and large repeats sub-diagonal) will also reduce variance.

3.3.3. Finding the Optimal Ordering for Larger Systems

There is no simple closed-form relationship which can determine the minimum length ordering of a 3x3 or larger matrix. One must either check all possible orderings if the optimal solution is desired, or use a
heuristic to check a subset which is likely to contain a good ordering. This subsection describes the optimization problem, while the next subsection gives a heuristic which can be used for the problem. The optimization scheme is an exponential-time algorithm.

The Markov chain framework can be used to evaluate and compare feasible orderings of the coupled tasks. The expected time for each candidate ordering is calculated as shown in section 3.2 above to determine the effect of the ordering on the time to complete the process.

Each of the $n!$ orderings (of an $n \times n$ matrix) can be checked in order to determine which is the minimum length (optimal) ordering using the following algorithm. The optimal ordering indicates the sequence in which the tasks should be attempted which will minimize the number and length of iterations which are necessary to fulfill the specifications.

**Optimization Algorithm:**

**Step 1:** Set initial ordering and evaluate.

Store the length of this initial ordering into best-ordering.

**Step 2:** Increment ordering.

Use any search technique such that the algorithm will cycle through all orderings. If there are no orderings remaining then Stop.

**Step 3:** Evaluate current ordering.

If the length of the ordering is better than previously stored best-ordering, then save as new best-ordering.

**Step 4:** Go back to Step 2.

A computer program was constructed which can evaluate the expected length of each possible ordering in order to find the one of
minimum length. This complete enumeration scheme is guaranteed to produce the optimal solution, but the length of time that it takes to complete grows exponentially with the size of the problem. The current encoding can find the optimal solution for up to an 8x8 system, but it takes approximately 10 hours to find the optimal solution of an 8x8 system on a Macintosh SE.

The 8x8 coupled system used in our test was the largest coupled block from the example in Steward's paper [Steward 1981]. As an example of the Design Structure Matrix technique, he describes the design dependencies for a hypothetical electric car design project. He gives a binary matrix which contains those dependencies and shows its partitioned state. The Design Structure Matrix for this design problem contains one large coupled block of eight tasks (out of 16 total tasks.) (The task names and numbers are those assigned by Steward.)

The coupled submatrix is shown in Figure 7. I have replaced each of the binary entries in the original matrix with a value that estimates the strength of the dependency between the tasks. I have also furnished estimates for the task times, in arbitrary units. The numerical entries are also arbitrary, based on the names of the tasks. Note that the shorter duration tasks are specification tasks, where important parameters are set; longer duration tasks are the actual design and complex calculation tasks, which require a significant amount of time to be completed.
Figure 7. Unordered Coupled Submatrix

The ordering of the matrix which produces the minimum expected project length is shown in Figure 8.

Figure 8. Ordered Coupled Submatrix

The total expected length of the optimal ordering is 29.9724. (The lower bound sum of all independent task lengths is 20.)

The optimal ordering makes intuitive sense. It says, in essence, to set the overall specifications (such as total weight and speed) early in the process, and then to do the detailed structural and aerodynamic design. The design is then checked to see if the specifications have been met, and if not, either the specifications are altered or the design is redone. This logical design process sequence suggests that when the model incorporates
real design information, it can provide assistance when attempting to organize even a coupled design process.

A number of quantitative features of the optimal ordering can also be noted. The longer tasks are located at the end of the matrix. By organizing the design process along those lines one would expect to have to repeat the longer tasks a fewer number of times. The largest matrix elements are subdiagonal, which minimizes the probability of iteration. These observations are consistent with those made in section 3.3.1 when considering the 2x2 matrix.

3.3.4. Approximate Solutions Obtained by Heuristic Search

Searching all of the n! orderings for the shortest one rapidly becomes unwieldy as n grows. It would be useful to identify a heuristic which would find a good ordering without having to check all possible orderings. It was observed that improvements to the current best ordering were frequently those which preserved the majority of the sequence, while moving one task relative to the other tasks.

For example, if the current best ordering is

\[
D \ E \ A \ B \ F \ C \ G
\]

then candidate improvements would be of the form

\[
D \ A \ B \ F \ E \ C \ G
\]

(where task E has been moved and all other tasks have remained in sequence.) This observation allows us to construct a local-search heuristic algorithm.

**Heuristic Algorithm:**

Step 1: Set initial ordering and evaluate its length.

Store length of this ordering as best-ordering.
Step 2: Set mobile-task to task 1.

Step 3: Move mobile task to one place later in the ordering.

   If it has been returned to its original position then set the
next task as the mobile-task. If every task has been the mobile-
task since the last improvement to the ordering then Stop.

Step 4: Evaluate current ordering.

   If better than previously stored best-ordering, save as
new best-ordering and go back to Step 2. Otherwise go to Step 3.

There are $n$ tasks which can be moved, and $n$ places to which a task
can be moved; therefore there are $n^2$ moves available from any starting
sequence. Candidate reorderings are checked until none of the $n^2$ such
candidates shows an improvement over the current best ordering. The
final ordering is not guaranteed to be optimal, but it does seem likely from
empirical evidence that the ordering will be good (in the sense that its
expected length is close to the minimum length.)

That type of switch is reasonable. It is likely that there could be a
small group of tightly coupled tasks within the entire coupled group for
which the ordering has a greater effect on the optimal solution. The
heuristic will identify those groups early in the process, and preserve their
ordering once they have been put in sequence. It will then shift about the
other tasks to obtain remaining improvements which are possible through
reordering. Another heuristic which would be of similar computational
complexity, attempting to switch pairs of tasks, seems likely to be able to
identify the important groups, but less able to preserve them while
swapping the remaining tasks.
The heuristic was applied to the 8x8 matrix described above. For this matrix the optimal solution is known. Fifty initial orderings were generated randomly. The heuristic always stopped within 0.1% of the optimal solution, and frequently found the optimal ordering.

The heuristic algorithm is as a 2-opt scheme (because the search neighborhood has \( n^2 \) entries.) The heuristic is significantly faster than complete enumeration. (For the example 8x8 matrix the best heuristic solution is found within 2 minutes on the Macintosh SE, in contrast to the 10 hours it took to find the optimal ordering.)

In order to understand the limitations of the above heuristic, consider an example 3x3 matrix where it is possible that the heuristic method will stop at a suboptimal ordering.

![Matrix Diagram](image)

(a) Optimal Ordering
E(Completion Time) = 364.3

(b) Suboptimal Ordering
E(Completion Time) = 367.2

Figure 9.

Figure 9(a) shows the optimal ordering. The suboptimal ordering in Figure 9(b) has no improvement available through the heuristic search. It would take two movements of the type described in the heuristic to move from the suboptimal ordering to the optimal ordering. The optimal ordering is less than 1% shorter in expected time than the suboptimal ordering.

This matrix (and all other 3x3 matrices observed which exhibit this behavior) has a short initial task relative to the subsequent tasks, and an optimal ordering which has the largest repeat probabilities above the diagonal.
Rather than the proposed heuristic, it would be possible to use a randomized algorithm (such as simulated annealing) that might be more likely to terminate at the global optimum, incurring the cost of additional computation time. Further investigation into the improvements available through alternative heuristic methods has not been undertaken, but would be worthwhile if the model is to be developed further.

3.4. Discussion

The sequential iteration model is an interesting description of the progress of complex engineering design projects. However, it is not clear that the model accurately captures the behavior of engineering design organizations. The questions to be answered are then: Does the model have any predictive or descriptive value reflecting the actual occurrence of engineering design? If not, how can it be improved? This section discusses several concerns about our model.

One way on which the model does not match a typical engineering design organization is that the probabilistic determination of which tasks require repetition is not (for a real organization) an independent choice. During the model's iteration, at most one of the previous tasks will be chosen for repetition if the most recently completed task is incompatible with upstream tasks. The assumption says that no matter what interface problem was created by completing the most recent piece of the design puzzle, working on just one other piece of the puzzle might be able to resolve that problem (as opposed to necessarily having to repeat several earlier design tasks). This assumption does not seem unreasonable, as one would like each design task to create no more problems than it solves. It would,
however, take some verification in an actual design setting to determine when this assumption would be realistic or valid.

Also, in the model each of the times and the repeat probabilities is assumed to be known and unchanging from one iteration to the next. In actuality, one might expect that during the second and later design iterations, an engineer would be able to complete the design more rapidly than during the first attempt. To a first approximation, it should be possible to assume that the task times are constant.

It is not obvious whether the repeat probabilities would change, and if they do change, whether they would increase or decrease. With more experience, an engineer should be able to focus in on the crux of the problem, thereby reducing the repeat probability. But for later iterations, it may be the case that the extent of the interface problem is larger than originally anticipated. The technical problem is difficult to overcome, so the likelihood of repetition is increased. Which of the countervailing effects is correct or appropriate is difficult to determine, but to a first approximation, it is assumed that the probabilities are constant and known.

There are several extensions to the current probabilistic model which would remove some or all of the above assumptions. The first would allow or force tasks to occur simultaneously. Second, the model could permit task times to change from one iteration to another. Third, the model could have repeat probabilities which change from iteration to iteration. All of these possibilities would expand the generality of the model, but might prove difficult to handle analytically.

Another possible extension to the model would be to introduce resource constraints or introduce multiple simultaneous projects which must compete for the same resources. It might be possible to analyze such
a system with a queueing network formulation, although it would be significantly more complex than the current model. It would also be possible to have a more general model which is analyzed using simulation techniques.

The model described here does serve as a useful description of the engineering design process. It enables analysis of the effects of functional design coupling on the progression of the design process, and enables prediction of the time it takes to complete a design project. It should be explored further to see if its assumptions are appropriate in an engineering design setting, and to see what insights can be derived from our modeling approach which would be useful to managers.

Also, other avenues for modeling the design process should be explored. It should be possible to change some of the assumptions of the model and come up with a comparable model which may be of greater applicability or generality. Our analytical method should serve as the basis for further research into managing engineering design projects.

There is uncertainty inherent in any design process. Because the development of the technology cannot be predicted precisely in advance, the engineering tasks to be performed cannot be completely specified prior to undertaking the project. Nevertheless, there is a role for such a tool in managing complex design projects. First, having the designers or their managers aid in the construction of a model of their own behavior forces them to build a consensus as to what the important design tasks are, and how they will relate to the other tasks [Graham 1985]. Even if the model produces a schedule which does not correspond to the actual progress of the project, they will have derived benefit by considering the broader implications of their portion of the design on the entire process.
Also, the model is consistent with the criteria identified by Liberatore and Titus [1983] concerning the role that management science models can play in project management. It is unlikely that a manager will give over control of the project to the suggestions of any automated decision support system. Management science models are most relevant to managers when they present information to the manager which allows justification of managerial decisions. Our model has its greatest effect in identification of which are the important tasks to be completed early, and which are the tightly coupled tasks. The manager is then able to assign coupled tasks to the engineering staff in such a way that the project will have an increased chance of technical success.

3.5. Conclusion

This chapter describes a powerful extension to the Design Structure Method which is useful for minimizing engineering development time. The method is quite tractable, although at the expense of some possibly unrealistic assumptions about the progression of the design process. Despite the restrictions imposed by the model, it does provide some useful recommendations. The model demonstrates that it is better in a sequential iteration design process to attempt a task only after the execution of tasks upon which it is strongly dependent, and to attempt the long tasks late in the design process, so the longer tasks are performed a minimum number of times.
Chapter 4. The Work Transformation Model

This chapter develops a distinct quantitative interpretation of the Design Structure Matrix (DSM) known as the Work Transformation Matrix. The Work Transformation Matrix assumes that each information dependency in a DSM can be interpreted as a quantity of rework which must be redone during the iteration process. The Work Transformation Matrix technique is applied to two industrial design problems, brake system design and electronics module design, and it is shown that the technique is able to identify the main iteration features of design in those design environments. The technique is also applied to an experimental design environment where it is possible to more explicitly demonstrate the relationships between the iteration process and design time.

4.1. Introduction to the Work Transformation Matrix

This chapter develops a model which addresses some of the shortcomings of the Sequential Iteration Model discussed in Chapter 3. The new model suggests that each iteration in the design process takes a decreasing amount of time. Also, the probabilities represented by the off-diagonal entries in the matrix are replaced by quantities which are more easily identified by design practitioners. These changes enable this model to be applied to industrial design settings more readily.

This is a descriptive model, not an optimization model. The description developed below can be used by the design manager to analyze the design problem, to estimate how long the design process will take, and to determine which aspects of the design problem contribute to iteration time.
The development of the model described in section 4.2 is based on a linear algebraic model of design time. The interpretation of the model is based on the eigenstructure (eigenvalues and eigenvectors) of the governing matrix. Section 4.3 contains an example matrix where the model is applied and described.

The description of the application of the model to industrial design environments are in sections 4.4 and 4.5. Section 4.4 contains a description of brake system design which is based on several months of field work at the brake system design division of General Motors. Section 4.5 describes a similar project concerning electronics module design from work done with engineers at Digital Equipment Company. In both cases the observations of the design environments depended on informal discussions with design engineers, internal documentation, and interviews with engineers and their managers.

Section 4.6 contains the results of a further attempt to verify the predictions of the Work Transformation Model. This section describes the application of the model to an experimental design environment, where the both the design process and design result can be observed and repeated. In this environment it is seen that there is good correspondence between the descriptions of the design process implicit in the Work Transformation model of the problem and the observed design result.

4.2. Work Transformation Matrix Model Development

For the purposes of this analysis, it is assumed that each task creates a deterministic amount of rework for other tasks. Rework is necessary because the task originally was attempted with imperfect information
(assumptions), and the rework adapts the original solution to account for the modified information.

We use a transformed version of a fully coupled Design Structure Matrix which we call the Work Transformation Matrix (WTM). The diagonal elements in the WTM represent the time that it takes to complete each task during the first iteration stage. (See Figure 1.) The off-diagonal elements represent strength-of-dependence measures (defined explicitly in section 4.2.1). It is assumed that there will be multiple iteration stages, and that the time for each stage is a function of the amount of time spent working in the previous stage. We wish to find the sum of the times of all stages.

![Strengths of Dependency and Task Times Diagram]

**Figure 1. Work Transformation Matrix**

The derivation below is divided into three sections. In section 4.2.1 we describe the assumptions underlying the model. In section 4.2.2 we describe why the eigenvalues and eigenvectors of the WTM are relevant to the analysis of development time. In section 4.2.3 we describe how the eigenstructure of the matrix is interpreted.

**4.2.1. Work Transformation Model Assumptions**

The assumptions in this model are:

- All tasks are done in every stage – fully parallel iteration
- Rework is created based on a linear rule – as a % of work done in previous iteration stage
• The parameters in the matrix describing work transformation behavior do not vary with time

These assumptions allow us to use a linear algebraic analytical method on the WTM.

To develop the model, we first introduce the concept of the work vector \( u_t \). This is an \( n \)-vector, where \( n \) is the number of design tasks to be completed. Each element of the work vector contains the amount of work to be done on each task after iteration stage \( t \). The initial work vector \( u_0 \) is a vector of ones, which indicates that all of the work remains to be completed on every task at the beginning of the iteration process.

During each iteration stage all work is completed on all of the design tasks. (For a relaxation of this assumption, where a fraction of the work is completed in every stage see Appendix 4.A.) However, work on a task will cause some rework to be created for all other tasks which are dependent on that task for information. We determine which tasks those are from the design structure matrix. Every iteration stage produces a change in the work vector according to:

\[
u_{t+1} = Au_t\]

where each of the entries \( a_{ij} \) in \( A \) implies that doing one unit of work on design task \( j \) creates \( a_{ij} \) units of rework for design task \( i \). The matrix \( A \) is obtained from the work transformation matrix by giving the off-diagonal elements this interpretation and setting the diagonal elements to zero. The work vector \( u_t \) can be also be expressed by:

\[
u_t = A^t u_0\]
The sum of each of the work vectors is the total work vector \( U \), the total number of times that each of the tasks is attempted during the total of \( T \) iteration stages of design process:

\[
U = \sum_{t=0}^{T} u_t
\]

or:

\[
U = \sum_{t=0}^{T} A^t u_0 = \left( \sum_{t=0}^{T} A^t \right) u_0
\]

The model output \( U \) is therefore in nominal units of iteration for each task. (If element \( i \) in vector \( U \) is 1.6, then the design organization will have done 60% rework on task \( i \) in subsequent stages.) (For a time-based interpretation of the matrix \( A \) see Appendix 4.B.) We scale \( U \) by the task durations to obtain units of task times. If \( W \) is a matrix which contains the task times along its diagonal, then \( WU \) is a vector which contains the amount of time (in engineer-hours) that each task will require during the first \( T \) iteration stages.

4.2.2. Eigenvalue Decomposition

If \( A \) has linearly independent eigenvectors (the eigenvector matrix \( S \) is invertible\(^1\)) then we can decompose \( A \) into:

\[
A = S \Lambda S^{-1}
\]

where \( \Lambda \) is a diagonal matrix of the eigenvalues of \( A \), and \( S \) is the corresponding eigenvector matrix. (For \( S \) to be invertible it is sufficient, but not necessary, that none of the eigenvalues be repeated.) The powers of \( A \) can be found by:

---

\(^1\) If \( A \) does not have linearly independent eigenvectors then \( A \) must be decomposed into its Jordan form. For more information on the Jordan form, please see appendix 4.C.
\[ A^t = S \Lambda^t S^{-1} \]

The total work vector \( U \) can therefore be expressed as:

\[ U = S \left( \sum_{t=0}^{T} \Lambda^t \right) S^{-1} u_0 \]

If the magnitude of the maximum eigenvalue is less than one, then the design process will converge (i.e. as \( T \) increases to infinity the total work vector \( U \) remains bounded.) An eigenvalue greater than one corresponds to a design process where doing one unit of work at some task during an iteration stage will create more than one unit of work for itself at some future stage. Such a system is unstable and the vector \( U \) will not converge, instead growing without bound as \( T \) increases. (It is a sufficient, but not necessary, condition for stability that the entries in every row sum to less than one.)

A design process which does not converge would be one where there is no technically feasible solution to the given specifications, or one where the designers are not willing to compromise to find the technical solution. This situation is not likely in the design environments we are modeling, that is, routine design where the designers are responsible for bringing out a new variation of a known successful product. The remainder of the discussion on Work Transformation Matrices is limited to problems where a technical solution exists and can be found in finite time (eigenvalues are less than 1.)

4.2.3. Interpreting the Eigenstructure

The eigenvalues and eigenvectors of matrix \( A \) determine the rate and nature of the convergence of the design process. Much can be learned about
what controls the iteration by looking at the eigenvalues and eigenvectors as opposed to looking at the sequence of work vectors.\(^2\)

A design mode is defined as a group of design tasks which are very closely related, and working on any one of them creates significant work, directly or indirectly, for each of the other tasks within the mode. It is possible to identify the design modes by looking at the eigenvalues and eigenvectors of matrix A.

The magnitude of each eigenvalue of A identifies the rate of convergence of each design mode. The eigenvector corresponding to each eigenvalue characterizes the relative contribution of each of the various tasks to the body of work which converges, as a group, at a given rate.

By the Perron-Frobenius Theorem (a fundamental result of matrix theory) we know that the largest magnitude eigenvalue of a coupled non-negative matrix will be real and positive [Marcus and Minc 1964]. Also, the eigenvector associated with this eigenvalue will have positive elements. (For a more complete discussion of the Perron-Frobenius Theorem see Appendix 4.C.)

The slowest design mode (largest eigenvalue) will therefore have an eigenvector which is strictly positive. This design mode gives us little problem with interpretation. Other design modes are, however, less obvious. Also by the Perron-Frobenius Theorem, there is only one eigenvector which is strictly positive. We must be able to interpret negative

\(^2\) The interpretation of the eigenvalues and eigenvectors for design problems is similar to the eigenstructure analysis used to examine the dynamic motion of a physical system. In the discrete time description of linear dynamic systems, each eigenvalue corresponds to a rate of convergence of one of the modes of the system (a natural frequency.) The eigenvectors identify the mode shapes of natural motion, quantifying the participation of each of the state variables in each mode [Ogata 1967].
and complex numbers in the eigenvectors as well as negative and complex eigenvalues.

Recalling that the total work vector $U$ is calculated by:

$$ U = S \left( \sum_{t=0}^{T} \Lambda^t \right) S^{-1} u_0 $$

we will look at each of the elements in the above formula for $U$ to see how the eigenstructure of matrix $A$ can be used to interpret the design modes. If we take the limit as $T$ approaches infinity we can use the formula:

$$ \lim_{T \to \infty} \sum_{t=0}^{T} \Lambda^t = (1 - \Lambda)^{-1} $$

If the maximum eigenvalue is not close to one, then the limit will be approached within relatively few iterations. For the remainder of this discussion the limit will be used, although the analysis can also be completed for finitely many iterations.

This limit is also a diagonal matrix, where each entry along the diagonal corresponds to one eigenvalue and has the form:

$$ \frac{1}{1 - \lambda} $$

where $\lambda$ is an eigenvalue.

In the next two subsections, both real and complex eigenvalues are discussed. In subsection 4.2.3.3 the interpretation of eigenvectors is considered.

4.2.3.1. **Real Eigenvalues**

The function:

$$ \frac{1}{1 - \lambda} $$
is strictly increasing over \((-1, 1)\). The graph of this function is shown in Figure 2.

**Figure 2. Graph of Magnitude vs. \(\lambda\) for Real Eigenvalues**

We see that all positive eigenvalues have the greater contribution to the series sum than do negative eigenvalues. Therefore, as we consider which are the more important design modes, we restrict our attention among real eigenvalues to the positive eigenvalues.

4.2.3.2. Complex Eigenvalues

For complex eigenvalues we also wish to find the magnitude of the limit of the sum of the infinite series. For a complex eigenvalue \(\alpha + \beta i\) the magnitude of the limit is:

\[
\left| \frac{1}{1 - (\alpha + \beta i)} \right| = \frac{1}{\sqrt{(1 - \alpha)^2 + \beta^2}}
\]

Or, alternatively:
\[
\left| \frac{1}{1 - (\alpha + \beta i)} \right| = \frac{1}{\sqrt{1 - 2\alpha + \alpha^2 + \beta^2}}
\]

Which, using the fact that:

\[\beta \neq 0\]

allows us to find an upper bound on the limit:

\[
\left| \frac{1}{1 - (\alpha + \beta i)} \right| < \frac{1}{1 - \alpha}
\]

Also, we can find a lower bound using the fact that:

\[\alpha^2 + \beta^2 < 1\]

to show:

\[
\left| \frac{1}{1 - (\alpha + \beta i)} \right| > \frac{1}{\sqrt{2 - 2\alpha}}
\]

The graph of the upper and lower bounds is shown in Figure 3.

---

**Figure 3. Graph of Bounds on Magnitude vs. \(\alpha\) for Complex Eigenvalues**
We see that the real part of complex eigenvalues gives bounds on the magnitude of the sum of the infinite series corresponding to that eigenvalue. We also see that complex eigenvalues with negative real part are not going to contribute significantly to the sum, and can therefore be ignored.

By the previous argument we need only consider those real eigenvalues which are positive. We therefore need to consider only those eigenvalues which have a positive real component, whether they are real or complex.

4.2.3.3. The Eigenvectors

This section discusses how the relative importance of each task within an eigenvectors is interpreted, given that we know the eigenvalue corresponding to that eigenvector. We want to be able to interpret the eigenvectors so that we can distinguish which of the tasks are important contributors to each design mode.

Again, consider the formula:

\[ U = S \left( \sum_{t=0}^{T} \Lambda^t \right) S^{-1} u_0 \]

We see that the final two terms in this formula:

\[ S^{-1} u_0 \]

give a weight for each eigenvector which is both a magnitude and a direction.

The eigenvector corresponding to real eigenvalues is real. Each weight for a real eigenvector is also real. Therefore, the direction is either positive or negative. The important quantities in a real eigenvector are
therefore the large positive values if the weight is positive, and large
negative values if the weight is negative.

Complex eigenvalues have complex eigenvectors and complex
weights. Determining how the direction of the weight and the direction of
the eigenvector interact is difficult. The best way to look at the interaction is
to calculate the contribution of the mode to the total work vector $U$ and see
which the tasks give large contribution to the total work.

Positive eigenvalues correspond to non-oscillatory design modes.
Negative and complex eigenvalues describe damped oscillations.
Oscillatory design modes indicate that the work is not decreasing for all of
the tasks in the mode at the same rate, but that the work is shifting from
task to task during iteration process.

The magnitude of the variability in the amount of work between
separate work vectors is not as important as the total magnitude of work
completed. The specifics of the variability would be useful if we were
tracking the individual task work information. Instead we are looking at
aggregate information, so the individual variability (as indicated by the
non-positivity of the eigenvector or eigenvalue) is less important. As we
interpret the modes of the design process we must therefore concentrate on
those modes with large positive real eigenvalues, or imaginary eigenvalues
with a large positive real part.

An illustration of the interpretation of the eigenvalues and
eigenvectors is given in the next section, where an example problem is fully
worked.
4.3. A Simple Example

As an illustration of the above discussion, let us consider the following 4x4 Work Transformation Matrix. This is a quantitative version of the coupled block (tasks C-F) in the camera design matrix as shown in Figure 3 of Chapter 2. The numbers can be interpreted as follows: if the shutter is completely redesigned, then 30% of the viewfinder design work must be redone (and so forth).

\[
A = \begin{bmatrix}
0 & 0.1 & 0.2 & 0.3 \\
0.3 & 0 & 0.4 & 0.2 \\
0.1 & 0.3 & 0 & 0.5 \\
0.1 & 0.1 & 0.2 & 0
\end{bmatrix}
\]

The eigenvalue (\(\Lambda\)) and eigenvector (\(S\)) matrices are:

\[
\Lambda = \begin{bmatrix}
0.674 & & & \\
& -0.392 & & \\
& & -0.141+0.060i & \\
& & & -0.141-0.060i
\end{bmatrix}
\]

\[
S = \begin{bmatrix}
0.410 & -0.067 & 0.657 & 0.657 \\
0.624 & -0.613 & 0.060 - 0.570i & 0.060 + 0.570i \\
0.580 & 0.758 & -0.395 + 0.073i & -0.395 - 0.073i \\
0.326 & -0.213 & -0.065 + 0.274i & -0.065 - 0.274i
\end{bmatrix}
\]

The four eigenvectors are the columns in \(S\). Each vector in \(S\) has as its eigenvalue the associated diagonal element of \(\Lambda\). The eigenvectors are (arbitrarily) scaled to be unit vectors. By inspection of the eigenvectors, we learn that the most slowly converging design mode (the one with the largest magnitude eigenvalue) involves primarily the middle two tasks. When we compute the first few work vectors, we find that they support the above interpretation.

\[
\begin{aligned}
u_0 &= \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} & u_1 &= \begin{bmatrix} 0.6 \\ 0.9 \\ 0.4 \end{bmatrix} & u_2 &= \begin{bmatrix} 0.39 \\ 0.62 \\ 0.53 \end{bmatrix} & u_3 &= \begin{bmatrix} 0.267 \\ 0.395 \\ 0.390 \\ 0.207 \end{bmatrix} & u_4 &= \begin{bmatrix} 0.180 \\ 0.278 \\ 0.249 \\ 0.144 \end{bmatrix}
\end{aligned}
\]
The work done on the first and last tasks is less than the work on the middle two tasks during all iteration stages. We see that the dominant mode describes the shape of convergence of the work vectors.

We can see what is happening by looking in more detail at the intermediate calculations used to determine the total amount of work completed during the iteration process. The infinite sum of the eigenvector matrix shows that the one positive eigenvector will contribute significantly more work to the process than the negative and the complex modes:

\[
(1 - \Lambda)^{-1} = \begin{bmatrix}
3.065 & 0.718 \\
0.874 + 0.046i & 0.874 - 0.046i
\end{bmatrix}
\]

The term used to see how each of the modes is represented in the original state vector is:

\[
S^{-1}u_0 = \begin{bmatrix}
2.125 \\
0.082 + 0.461i
\end{bmatrix}
\]

Multiplying this weighting vector by the sum of the eigenvalue matrix we find the total weight on the eigenvector matrix:

\[
(1 - \Lambda)^{-1}S^{-1}u_0 = \begin{bmatrix}
6.513 \\
0.093 - 0.399i
\end{bmatrix}
\]

Note that the weight on the first eigenvector is significantly larger in magnitude than the other weights. Most of the work in this iteration process is described by the primary design mode.

We are now able to calculate the total work vector:

\[
U = S(1 - \Lambda)^{-1}S^{-1}u_0 = \begin{bmatrix}
2.807 \\
3.755 \\
3.595 \\
2.375
\end{bmatrix}
\]
There has been more work completed during the process by the middle two tasks, just as the preliminary analysis of the eigenvectors and eigenvalues indicated.

4.4. Brake System Design

This section is intended to verify the utility of the Work Transformation technique by demonstrating the analysis of an actual design process and showing the insights gained. A design structure matrix for the brake system was reported previously [Black 1990, Black et al. 1990.] The work described here applied the Work Transformation Matrix method (described in section 4.2) to the brake system design process. In preparing this analysis, the author spent several months doing field work at the brake system design facility of General Motors. That work included interviews with brake system engineers at several sites.

There are four questions which must be answered in constructing the Design Structure Matrix. We must first determine all of the various steps or tasks in the design process. Second, we must determine all of the information flows between the various tasks. Third, we must determine the relative importance of each of the information flows (quantifying the off-diagonal elements in the matrix). Fourth, we must estimate the time it takes to complete each task.

The brake system DSM from Black et al. [1990] is shown in Figure 4a. The matrix demonstrates a problem which can be divided into a block of complex, coupled design parameters at the center of the matrix, preceded by and followed by a group of sequential and parallel parameters. The coupled block is expanded in Figure 4b. (We realize that Figure 4a is too
small to see the details of the matrix; it is included to demonstrate the overall structure of the DSM, which includes over 100 design parameters.)

(a) Complete Matrix

(b) Expansion of Iterative Block

Figure 4. Brake System Design Matrix

The interactions between design parameters are poorly understood, which leads to the coupling and the complexity of the design. If all interactions were well understood on a technical level, then the behavior could be described by predictive mathematical models (analytical or simulation). Iteration using such predictive models would be relatively fast. However, since there are many system level interactions which are not well enough understood to create a good predictive model, there are many lengthy iterations in the brake system design process. These iterations include costly and time-consuming experiments.

4.4.1. The Engineer's View

The customer wants an automobile with quiet, smooth brakes that do not require frequent service. To the engineer this means that the designed brake system should have little or no brake squeal or brake pulsation, and
that the linings should have a long life. These problems are known as noise, pulsation and wear. The generic causes of inability to meet these functional requirements are understood by engineers – stick-slip friction excites audible resonances in the rotor and other nearby structures, uneven rotor wear leads to pulsation, and elevated lining temperature leads to rapid wear of the brake linings. More specific causes remain unknown. Detailed analysis of these problems continues, and some progress is being made. The sentiment among engineers is that none of these problems will be 'solved' in the near future. Specifically, brake systems cannot be designed so that no customers ever complain about these three problems. These problems are believed to be inherent consequences of using dry friction to stop a vehicle.

These three problems (noise, pulsation and wear) have been identified by designers and their managers as the 'controlling features' of the design/test/redesign iteration problem they experience. As shown below, the WTM analysis confirms that these are indeed controlling issues in design iteration, and details the specific contributing parameters for each. The match between designer perception and analytical identification lends credence to both approaches.

4.4.2. Using the Work Transformation Method to Identify Iteration Drivers

Using the analytical tools described in section 4.2, we can more rigorously identify the parameters within the large coupled block which compose the most interrelated sets of parameters. The original DSM analysis of the brake system identified parameters within the large block [Black 1990]. This work furthers the analysis by recognizing that some of the parameters exhibit stronger interdependence than others, and that tightly coupled parameters consequently require more iteration during the
design process. The dominant design modes would be interpreted as 'controlling features' or 'design drivers', in that they require more engineering time during the design process and they are likely to be on the critical path of the design project.

To perform this analysis, we translate the binary DSM (Figure 4b) into a Work Transformation Matrix. In lieu of precise numerical values in the Work Transformation Matrix for the brake system, the individual cells were estimated to be of either weak, medium, or strong dependence. (See Figure 5.) These values are based on the author's perceptions of how much rework each upstream task creates for the downstream tasks based on experience at the field site. We have used the values 0.5, 0.25, 0.05 for strong, medium, and weak dependence, respectively. Our experience shows that the identification of the design drivers is robust against minor changes in the values entered in the matrix. (See Appendix 4.E for a discussion of sensitivity of the eigenvectors to changes in the weights.)
|   | 23 | 24 | 35 | 36 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 104 |
|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 33 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Strong Dependence |
| 34 |    |    |    |    | X  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Medium Dependence |
| 35 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |
| 36 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |
| 37 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |
| 38 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |
| 39 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |
| 40 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |
| 41 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |
| 42 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |
| 43 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |
| 44 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |
| 45 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |
| 46 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |
| 47 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |
| 48 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |
| 49 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |
| 50 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |
| 51 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |
| 52 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |
| 53 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |
| 54 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |
| 55 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |
| 56 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |
| 57 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |
| 58 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |
| 59 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |
| 60 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |
| 61 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |
| 62 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |
| 63 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |
| 64 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |
| 65 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |
| 66 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | Weak Dependence    |

**Figure 5. Coupled Block from Brake Matrix with Weighted Dependencies**

The dominant design modes are represented by the eigenvector of the work transformation matrix corresponding to the largest eigenvalues. Figure 6 shows a graph of the magnitudes of all the eigenvalues. The graph shows their relative magnitude, which is an indicator of relative number of iteration stages until they are completed. Table 1 shows the values of the elements in the first two eigenvectors. (The larger magnitude elements are highlighted for emphasis and the eigenvectors are scaled to be unit vectors.)
Figure 6. Brake System Eigenvalues
<table>
<thead>
<tr>
<th>#</th>
<th>Parameter</th>
<th>First</th>
<th>Second</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>Knuckle envelope &amp; attach pts</td>
<td>0.029</td>
<td>0.109</td>
</tr>
<tr>
<td>34</td>
<td>Pressure at rear wheel lock up</td>
<td>0.305</td>
<td>-0.065</td>
</tr>
<tr>
<td>35</td>
<td>Brake torque vs. skidpoint</td>
<td>0.257</td>
<td>-0.014</td>
</tr>
<tr>
<td>37</td>
<td>Line pressure vs. brake torque</td>
<td>0.125</td>
<td>-0.002</td>
</tr>
<tr>
<td>40</td>
<td>Splash shield geometry—front</td>
<td>0.099</td>
<td>0.437</td>
</tr>
<tr>
<td>44</td>
<td>Drum envelope &amp; attach pts</td>
<td>0.004</td>
<td>0.023</td>
</tr>
<tr>
<td>45</td>
<td>Bearing envelope &amp; attach pts</td>
<td>0.033</td>
<td>0.132</td>
</tr>
<tr>
<td>46</td>
<td>Splash shield geometry—rear</td>
<td>0.021</td>
<td>0.084</td>
</tr>
<tr>
<td>48</td>
<td>Air flow under car/wheel space</td>
<td>0.047</td>
<td>0.326</td>
</tr>
<tr>
<td>49</td>
<td>Wheel material</td>
<td>0.007</td>
<td>0.054</td>
</tr>
<tr>
<td>50</td>
<td>Wheel design</td>
<td>0.02</td>
<td>0.103</td>
</tr>
<tr>
<td>51</td>
<td>Tire type/material</td>
<td>0.078</td>
<td>-0.007</td>
</tr>
<tr>
<td>52</td>
<td>Vehicle deceleration rate</td>
<td>0.58</td>
<td>-0.087</td>
</tr>
<tr>
<td>53</td>
<td>Temperature at components</td>
<td>0.106</td>
<td>0.168</td>
</tr>
<tr>
<td>54</td>
<td>Rotor cooling coefficient</td>
<td>0.102</td>
<td>0.464</td>
</tr>
<tr>
<td>55</td>
<td>Lining—rear vol and area</td>
<td>0.096</td>
<td>0.006</td>
</tr>
<tr>
<td>56</td>
<td>Rotor width</td>
<td>0.101</td>
<td>0.499</td>
</tr>
<tr>
<td>57</td>
<td>Pedal attach pts</td>
<td>0.113</td>
<td>-0.118</td>
</tr>
<tr>
<td>58</td>
<td>Dash deflection</td>
<td>0.174</td>
<td>-0.154</td>
</tr>
<tr>
<td>59</td>
<td>Pedal force (required)</td>
<td>0.414</td>
<td>-0.178</td>
</tr>
<tr>
<td>60</td>
<td>Lining material—rear</td>
<td>0.123</td>
<td>-0.054</td>
</tr>
<tr>
<td>61</td>
<td>Pedal mechanical advantage</td>
<td>0.219</td>
<td>-0.122</td>
</tr>
<tr>
<td>62</td>
<td>Lining—front vol &amp; swept area</td>
<td>0.114</td>
<td>0.088</td>
</tr>
<tr>
<td>63</td>
<td>Lining material—front</td>
<td>0.28</td>
<td>0.007</td>
</tr>
<tr>
<td>64</td>
<td>Booster reaction ratio</td>
<td>0.198</td>
<td>-0.066</td>
</tr>
<tr>
<td>65</td>
<td>Rotor diameter</td>
<td>0.121</td>
<td>0.024</td>
</tr>
<tr>
<td>66</td>
<td>Rotor envelope &amp; attach pts</td>
<td>0.01</td>
<td>0.057</td>
</tr>
<tr>
<td>104</td>
<td>Rotor material</td>
<td>0.066</td>
<td>0.209</td>
</tr>
</tbody>
</table>

Can we stop the car? Will the brake system overheat?

Table 1. Brake System Eigenvectors
The first design mode is primarily composed of Vehicle Deceleration Rate and Pedal Force Required, with lesser input from Pressure at Rear Wheel Lockup, Front Lining Material, Booster Reaction Ratio, Dash Deflection, Brake Torque vs. Skidpoint, and Pedal Mechanical Advantage.

What this design mode shows is that the group of design parameters which requires the greatest number of design iterations before convergence on the final acceptable design is the stopping distance problem, represented by the first column in Table 1. Solving this problem assures that the brake system is going to stop the car without creating uncontrollable skidding. A performance simulation for this problem exists, and it is a good predictor of actual performance. These iterations can therefore occur quickly. The large number of iterations on the first design mode does not strongly affect the total time of the development process. This model nevertheless shows that the stopping performance problem is the fundamental controlling feature which affects design iteration.

The second design mode is composed of primarily Rotor Cooling Coefficient, Airflow under Wheel Space, Splash Shield Geometry and Rotor Width, with lesser input from Rotor Material and Temperature at Components. All of these factors are technical parameters corresponding to overheating and cooling. This second design mode, which is composed of the cooling coefficient/rotor material problem, is related to the problems of lining life, noise generation, and pulsation problems. For these 'thermal' problems, there are few analytical or simulation tools available to the designer. Many iterations are required to converge upon a design solution, but there is no guarantee that those iterations can be rapid. Field or laboratory testing must be used to eventually converge on a solution which meets the criteria at a relatively high cost in time.
The design modes analysis has been able to identify the dominant controlling features correctly. This success is made evident by the engineers’ *a priori* prediction that noise, pulsation and wear would be found to be the fundamental design issues. We not only confirmed this, but also described these problems more precisely and showed how coupled these issues are.

4.4.3. Comparison with Shunt Diagram Method

In order to demonstrate the insights which are available through the WTM approach, we will compare those results with one other method used to analyze coupled blocks. In this method we attempt to decompose the coupled block in the brake system using the hierarchical, shunt diagram approach of Steward. (See Section 2.3.1 for a discussion of shunt diagrams.) The shunt diagram method attempts to move the most heavily-weighted dependencies below the diagonal, repeating with each level of weighting. Figure 7 contains the matrix which is obtained after hierarchical reordering.
Figure 7. Reordered Brake System Matrix

What we observe in the reordered matrix is that the two main design modes do appear, although in a slightly different form. There is a group of twelve tasks where the medium and strong elements cannot be made subdiagonal. Within this block, there are two blocks of six elements which are even more tightly grouped. These two blocks of six elements correspond to the first and second design modes identified through the WTM technique.

There are, however, three tasks which appear in the primary design mode which are not within that coupled block. These are Brake Torque vs. Skidpoint, Pressure at Rear Wheel Lock Up, and Vehicle Deceleration Rate. The hierarchical reordering, in an attempt to find a sequence which will lessen the need to go back and do previous tasks again, recommends that
these three tasks be done later the design process. Since there is a discrepancy between the two ideas, we have to look at the assumptions of the Work Transformation Model to see what causes this misfit.

The Work Transformation Model assumes that all tasks will be done during every iteration stage. If that assumption is correct, then the previous algebraic analysis shows that the tasks in the main design modes will have a greater amount of rework during the design process.

However, if some tasks are delayed until later in the process, less rework on the later tasks may be required. The cost of this delay might be an overall lengthening of the design process. It would be useful if it was possible to identify when this choice is appropriate. In order for the reduction in rework to have value, it must be assumed that a reduction in rework reduces engineering costs, such that either the total number of engineers can be reduced or else the engineers can be usefully employed on other projects while they are waiting to begin work on this project. The model developed in section 4.2 does not allow delaying some tasks, since the model assumes a fully simultaneous design process. A model which addresses delaying some design tasks in order to reduce iteration is discussed in section 4.5, where module design is presented.

4.4.4. What Has Been Learned by Modeling Brake System Design

This section has applied the Work Transformation Matrix to an actual design process. The analysis demonstrates the utility of eigenvalues and eigenvectors of the Work Transformation Matrix in interpreting the amount of work which must be completed during the design iteration process. The eigenvectors can be used to identify the 'controlling features', those elements of a coupled design problem which require the greatest number of iterations to reach a technical solution.
The Work Transformation Matrix can serve as a useful diagnostic tool in analyzing coupled design problems. We believe that this analytical method can lead to improvements in design processes by focusing attention on the slowly converging design iteration modes. For the brake system design process we postulate that improved simulation of the thermal and vibrational aspects of the design problem may accelerate solution development. Verification that an improved thermal simulation would reduce the amount of time that it takes to complete the brake system design process remains as future work.

4.5. Electronics Module Design

Section 4.4 described an application of the Work Transformation Matrix method to brake system design. This analysis was able to identify the main sources of iteration in that design environment. It was desirable to see if it was possible to repeat this success on a technology which was less familiar, in a way which would reduce the potential for bias in the data caused by the modeler’s familiarity with the technology at hand.

An electronics module is an assembled circuit board which contains integrated circuits and other electronics components. Modules are used in all electronics, from portable radios to high performance computers. The design and manufacture of modules are crucial aspects of the success of computer manufacturing.

Computers which use state-of-the-art technology have very complex modules. Currently, this means printed circuit board with many layers, lines on the circuit board which are very thin, surface mount components with leads of a fine pitch, and tight manufacturing and performance tolerances. The modules being studied were used on high performance
computer workstations at the Digital Equipment Company, where there are tight tolerances and large competitive pressures.

As we consider the module design process, we include the design of both the product and the manufacturing process. The product design includes the logical design of the circuit and the design of the printed circuit board. The manufacturing process includes the fabrication of the printed circuit board and the design of the assembly process used to produce the completed module. Module design provides an environment which demonstrates the applicability of the WTM method to process as well as product design, and which shows some extensions to the model which increase its ability to respond to the different type of design environment.

4.5.1. Goals of Module Design Modeling

One criticism of the brake system design modeling effort described in section 4.4 is that the matrix model is quite specific to that design environment and that there is much implicit knowledge of the design process embedded in the matrix information. The module design modeling effort was undertaken to demonstrate that the insights into the iteration process available through using the WTM modeling technique could be transferred to other technical environments.

There are two types of data gathered from the organization: the tasks and dependencies which are used to construct the design structure matrix, and the descriptions of iteration which are used to verify that the work transformation model is accurately capturing iterative behavior. It was the goal of this research to gather these types of data in a way which lessens the chance for bias. The reduction in bias is accomplished by gathering the two types of data from different sources. The matrix data and their collection
are discussed in sections 4.5.2 and 4.5.4, while the iteration description data are discussed in 4.5.7.

Also, there was a greater reliance on direct data collection, such as having engineers in the organization set the weights on the dependencies in the matrix. This step reduces the introduction of the researcher's biases in the collection and interpretation of the data.

4.5.2. The Module Design Data

We started building a design matrix based on an internal report dealing with information flow in module design. The report was compiled in 1989 so that managers of the process could document and monitor information flows in the design process. The report described module design as a generic process, and the technical information remains current.

The technical report uses the SADT modeling technique as described in section 1.4 [Ross 1977]. The SADT method is used to describe a hierarchical description of information flows through a technical organization. In building the report, the information flows are documented through a series of interviews with the responsible engineers.

4.5.3. How the Data were Turned into Matrix Form

We used the information flows described in the SADT report to build a design structure matrix. Tasks in an SADT report are arranged hierarchically. Higher level tasks are divided into lower level tasks. The DSM method is a flat description of task interrelationships, and therefore requires all tasks at the same level of hierarchy.

Each lowest level task in the report was turned into a task (row and associated column) in the matrix. The report does not divide all highest
level of task into the same number of sub-levels. All tasks which are represented in the charts in this thesis are the lowest level of task description provided by the report.

Each arc in the SADT graph was turned into an entry in the matrix. The arc is traced from its source lowest-level task to its destination lowest-level task, although the arcs often extend across different pages and different levels of the report. This is a straightforward but time-consuming procedure, as it requires tracing hundreds of arcs over several score pages of the report in order to find their sources and destinations.

4.5.4. Setting the Weights in the Matrix

We asked engineers at Digital who were familiar with module design to assign weights to the entries in the matrix. The weights indicate the level of importance of the given input information to the task at hand. They specified three levels of information importance: high, medium and low.

High-level information is critical to the completion of the design task. Without this information it is extremely difficult to produce a good design choice during that design step.

Medium-level information is still important, although less critical. The designer may be able to make an educated guess without medium level information available, but the guess will need to be reconciled with information to be provided later.

Low-level information is of lesser importance. This is information which the designer would like to have available, although it is likely that the design can be made reasonably accurately even when this information remains unavailable.
4.5.5. Concurrent Design of Product and Process

First we will consider the case where product design and process design are done concurrently. Under the assumptions of the model described in section 4.2 all tasks are attempted during every iteration. Therefore, both product and process design tasks will be worked on from the beginning of the design process. The product design tasks are those under the A1 hierarchy, where task A1 is Design Module. The process design tasks are those under the A2 hierarchy, where task A2 is Develop Manufacturing Process.

4.5.5.1. The Eigenstructure

The eigenvectors corresponding to the largest two eigenvalues are given in Figure 8. As described in section 4.2, this eigenvector indicates where the greatest amount of work will be done during the iteration process. These vectors have been normalized to have unit length. The task weightings with large positive values are indicated in bold.
<table>
<thead>
<tr>
<th>Activity</th>
<th>Eigenvector First</th>
<th>Eigenvector Second</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1111 Choose Arch. &amp; Technology</td>
<td>0.069</td>
<td>0.100</td>
</tr>
<tr>
<td>A1112 Develop Module Function</td>
<td>0.069</td>
<td>0.155</td>
</tr>
<tr>
<td>A1113 Simulate Module Architecture</td>
<td>0.018</td>
<td>0.066</td>
</tr>
<tr>
<td>A112 Maintain Component Libraries</td>
<td>0.128</td>
<td>-0.088</td>
</tr>
<tr>
<td>A1131 Create Design</td>
<td>0.448</td>
<td>0.057</td>
</tr>
<tr>
<td>A1132 Capture Schematic</td>
<td>0.151</td>
<td>-0.013</td>
</tr>
<tr>
<td>A11331 Simulate &amp; Validate Logic</td>
<td>0.161</td>
<td>-0.015</td>
</tr>
<tr>
<td>A11332 Simulate &amp; Validate Timing</td>
<td>0.161</td>
<td>-0.015</td>
</tr>
<tr>
<td>A11333 Validate Logic &amp; Timing</td>
<td>0.161</td>
<td>-0.015</td>
</tr>
<tr>
<td>A11411 Translate Format</td>
<td>0.081</td>
<td>-0.012</td>
</tr>
<tr>
<td>A11412 Schedule Logical Connects</td>
<td>0.038</td>
<td>-0.024</td>
</tr>
<tr>
<td>A11413 Assign Design Codes</td>
<td>0.038</td>
<td>-0.024</td>
</tr>
<tr>
<td>A11414 Check Logic Design</td>
<td>0.038</td>
<td>-0.024</td>
</tr>
<tr>
<td>A11421 Place Parts on Cover</td>
<td>0.204</td>
<td>-0.074</td>
</tr>
<tr>
<td>A11422 Add/Optimize Connects</td>
<td>0.176</td>
<td>-0.131</td>
</tr>
<tr>
<td>A11423 Layout the Etch</td>
<td>0.163</td>
<td>-0.151</td>
</tr>
<tr>
<td>A11431 Run Layout vs. Mfg Rule</td>
<td>0.086</td>
<td>-0.093</td>
</tr>
<tr>
<td>A11432 Run Full Spacing Analysis</td>
<td>0.133</td>
<td>-0.011</td>
</tr>
<tr>
<td>A11433 Add Final Functionality</td>
<td>0.036</td>
<td>-0.034</td>
</tr>
<tr>
<td>A11434 Create Mfg Data Package</td>
<td>0.056</td>
<td>-0.052</td>
</tr>
<tr>
<td>A1151 Produce Raw Board Data</td>
<td>0.015</td>
<td>-0.022</td>
</tr>
<tr>
<td>A1152 Produce Asmbld Board Data</td>
<td>0.015</td>
<td>-0.022</td>
</tr>
<tr>
<td>A1153 Create Test Packages</td>
<td>0.062</td>
<td>-0.078</td>
</tr>
<tr>
<td>A12 Judge Productivity</td>
<td>0.177</td>
<td>0.193</td>
</tr>
<tr>
<td>A13 Build &amp; Test Module Prototype</td>
<td>0.143</td>
<td>0.219</td>
</tr>
<tr>
<td>A14 Develop Module Diagnostics</td>
<td>0.084</td>
<td>0.039</td>
</tr>
<tr>
<td>A15 Qualify Module Design</td>
<td>0.085</td>
<td>0.040</td>
</tr>
<tr>
<td>A211 Develop Overall Scheme</td>
<td>0.158</td>
<td>0.140</td>
</tr>
<tr>
<td>A2121 Design Details of Workstations</td>
<td>0.145</td>
<td>0.097</td>
</tr>
<tr>
<td>A2122 Build Workstation Prototype</td>
<td>0.038</td>
<td>0.042</td>
</tr>
<tr>
<td>A2123 Dvip Tool Generation Capabil</td>
<td>0.038</td>
<td>0.042</td>
</tr>
<tr>
<td>A2124 Test Workstation Prototype</td>
<td>0.050</td>
<td>0.077</td>
</tr>
<tr>
<td>A213 Develop Process Tests</td>
<td>0.066</td>
<td>0.112</td>
</tr>
<tr>
<td>A221 Construct and Lay Out Facility</td>
<td>0.076</td>
<td>0.146</td>
</tr>
<tr>
<td>A222 Install Workstations</td>
<td>0.096</td>
<td>0.209</td>
</tr>
<tr>
<td>A223 Train Operators</td>
<td>0.076</td>
<td>0.146</td>
</tr>
<tr>
<td>A2311 Develop Board Processes</td>
<td>0.224</td>
<td>0.253</td>
</tr>
<tr>
<td>A2312 Develop Assembly Processes</td>
<td>0.204</td>
<td>0.256</td>
</tr>
<tr>
<td>A2313 Develop Assy Mfg Tests</td>
<td>0.173</td>
<td>0.135</td>
</tr>
<tr>
<td>A2321 Gerate Board Softools</td>
<td>0.161</td>
<td>0.126</td>
</tr>
<tr>
<td>A2322 Make Board Fixtures</td>
<td>0.161</td>
<td>0.126</td>
</tr>
<tr>
<td>A2323 Generate Assembly Softools</td>
<td>0.231</td>
<td>0.280</td>
</tr>
<tr>
<td>A2324 Make Assembly Fixtures</td>
<td>0.152</td>
<td>0.128</td>
</tr>
<tr>
<td>A2325 Generate Assy Test Pgm</td>
<td>0.163</td>
<td>0.136</td>
</tr>
<tr>
<td>A233 Test Mfg Process</td>
<td>0.329</td>
<td>0.604</td>
</tr>
</tbody>
</table>

**Figure 8. Eigenvectors of Product and Process Matrix**
4.5.5.2. Interpreting the Eigenvectors

The first eigenvector in the product and process design matrix (Figure 8), or the eigenvector corresponding to the largest eigenvalue, is primarily composed of both product design and process design tasks. This eigenvector indicates that there will be much iteration between product design and assembly system design if the product and process design are done in parallel.

The second eigenvector in the matrix (the eigenvector corresponding to the second largest eigenvalue) is primarily composed of process design tasks. This design mode indicates that there will be significant iteration among the process design tasks.

We see that there will be a large amount of iteration which includes both product design and process design phases of the design process. The next section illustrates the effects of the iteration on the total amount of work completed during this design process.

4.5.5.3. Total Work Vectors

Using the formulas developed in subsection 4.2.2 we can calculate the total work completed during the design iteration process. The result of this calculation is shown in Figure 9, which is the total work vector $\mathbf{U}$ for the prescribed design strategy. (Each value in this vector is the number of times that each task will have to be attempted during the design process. For example, task A11423 Layout the Etch will be complete 5.86 times during the iteration process.) The numbers in bold are the tasks which are repeated more than 5 times. This indicates a significant amount of iteration. We see that most of these tasks are those which are highlighted in the eigenvectors in Figure 8 above.
<table>
<thead>
<tr>
<th>Task Code</th>
<th>Task Description</th>
<th>Time (hrs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1111</td>
<td>Choose Arch. &amp; Technology</td>
<td>2.99</td>
</tr>
<tr>
<td>A1112</td>
<td>Develop Module Function</td>
<td>3.15</td>
</tr>
<tr>
<td>A1113</td>
<td>Simulate Module Architecture</td>
<td>1.63</td>
</tr>
<tr>
<td>A112</td>
<td>Maintain Component Libraries</td>
<td>4.36</td>
</tr>
<tr>
<td>A1131</td>
<td>Create Design</td>
<td>12.90</td>
</tr>
<tr>
<td>A1132</td>
<td>Capture Schematic</td>
<td>4.45</td>
</tr>
<tr>
<td>A11331</td>
<td>Simulate &amp; Validate Logic</td>
<td>4.67</td>
</tr>
<tr>
<td>A11332</td>
<td>Simulate &amp; Validate Timing</td>
<td>4.67</td>
</tr>
<tr>
<td>A11333</td>
<td>Validate Logic &amp; Timing</td>
<td>4.67</td>
</tr>
<tr>
<td>A11411</td>
<td>Translate Format</td>
<td>2.83</td>
</tr>
<tr>
<td>A11412</td>
<td>Schedule Logical Connects</td>
<td>2.00</td>
</tr>
<tr>
<td>A11413</td>
<td>Assign Design Codes</td>
<td>2.00</td>
</tr>
<tr>
<td>A11414</td>
<td>Check Logic Design</td>
<td>2.00</td>
</tr>
<tr>
<td>A11421</td>
<td>Place Parts on Cover</td>
<td>6.92</td>
</tr>
<tr>
<td>A11422</td>
<td>Add/Optimize Connects</td>
<td>6.20</td>
</tr>
<tr>
<td>A11423</td>
<td>Layout the Etch</td>
<td>5.86</td>
</tr>
<tr>
<td>A11431</td>
<td>Run Layout vs. Mfg Rule</td>
<td>3.64</td>
</tr>
<tr>
<td>A11432</td>
<td>Run Full Spacing Analysis</td>
<td>4.87</td>
</tr>
<tr>
<td>A11433</td>
<td>Add Final Functionality</td>
<td>2.18</td>
</tr>
<tr>
<td>A11434</td>
<td>Create Mfg Data Package</td>
<td>2.89</td>
</tr>
<tr>
<td>A1151</td>
<td>Produce Raw Board Data</td>
<td>1.58</td>
</tr>
<tr>
<td>A1152</td>
<td>Produce Assembld Board Data</td>
<td>1.58</td>
</tr>
<tr>
<td>A1153</td>
<td>Create Test Packages</td>
<td>3.14</td>
</tr>
<tr>
<td>A12</td>
<td>Judge Productibility</td>
<td>6.15</td>
</tr>
<tr>
<td>A13</td>
<td>Build &amp; Test Module Prototype</td>
<td>5.22</td>
</tr>
<tr>
<td>A14</td>
<td>Develop Module Diagnostics</td>
<td>3.66</td>
</tr>
<tr>
<td>A15</td>
<td>Quality Module Design</td>
<td>3.79</td>
</tr>
<tr>
<td>A211</td>
<td>Develop Overall Scheme</td>
<td>6.29</td>
</tr>
<tr>
<td>A2121</td>
<td>Design Details of Workstations</td>
<td>5.71</td>
</tr>
<tr>
<td>A2122</td>
<td>Build Workstation Prototype</td>
<td>2.14</td>
</tr>
<tr>
<td>A2123</td>
<td>Devl Tool Generation Capabil</td>
<td>2.14</td>
</tr>
<tr>
<td>A2124</td>
<td>Test Workstation Prototype</td>
<td>3.00</td>
</tr>
<tr>
<td>A213</td>
<td>Develop Process Tests</td>
<td>3.29</td>
</tr>
<tr>
<td>A221</td>
<td>Construct and Lay Out Facility</td>
<td>3.62</td>
</tr>
<tr>
<td>A222</td>
<td>Install Workstations</td>
<td>4.35</td>
</tr>
<tr>
<td>A223</td>
<td>Train Operators</td>
<td>3.62</td>
</tr>
<tr>
<td>A2311</td>
<td>Develop Board Processes</td>
<td>8.00</td>
</tr>
<tr>
<td>A2312</td>
<td>Develop Assembly Processes</td>
<td>7.55</td>
</tr>
<tr>
<td>A2313</td>
<td>Develop Assy Mfg Tests</td>
<td>6.87</td>
</tr>
<tr>
<td>A2321</td>
<td>Gerate Board Softools</td>
<td>6.42</td>
</tr>
<tr>
<td>A2322</td>
<td>Make Board Fixtures</td>
<td>6.42</td>
</tr>
<tr>
<td>A2323</td>
<td>Generate Assembly Softools</td>
<td>8.45</td>
</tr>
<tr>
<td>A2324</td>
<td>Make Assembly Fixtures</td>
<td>6.19</td>
</tr>
<tr>
<td>A2325</td>
<td>Generate Assy Test Pgms</td>
<td>6.65</td>
</tr>
<tr>
<td>A233</td>
<td>Test Mfg Process</td>
<td>11.34</td>
</tr>
</tbody>
</table>

**Figure 9. Total Work Vectors for Concurrent Design**
4.5.6. Sequential Design of Product and Process

4.5.6.1. Why we Look at Blocks Separately

One of the assumptions of the work transformation model presented in section 4.2 is that tasks within each block are attempted simultaneously. Therefore, in order to apply the model, we must identify parts of the matrix which are being attempted simultaneously. For the module design process, product design is not done simultaneously with detailed process design. The details of process design, such as the layout of the equipment, the programming of the software of the assembly equipment, and other details, are highly dependent on the details of the design. In practice these tasks are not done concurrently with product design. Some of the larger manufacturing issues are considered during product design, such as setting overall line widths on the board, and developing any new generic manufacturing process technology.

At the time that the manufacturing facility is being developed, it is too late to change the product design. There are intense time pressures, and the manufacturing process will be adapted to accommodate the design as it is.

4.5.6.2. The Eigenstructure

Assuming that all of the product design tasks (those within the A1 hierarchy) are done simultaneously, we can find the eigenvalues and eigenvectors which describe the iteration for this design process.

Figure 10 shows the eigenvectors corresponding to the two largest eigenvalues. These design modes will require the greatest amount of work during the product design phase of the design process. (Again, larger numbers are highlighted in bold for emphasis.)
<table>
<thead>
<tr>
<th>Task</th>
<th>Eigenvectors First</th>
<th>Eigenvectors Second</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1111 Choose Arch. &amp; Technology</td>
<td>0.067</td>
<td>0.179</td>
</tr>
<tr>
<td>A1112 Develop Module Function</td>
<td>0.062</td>
<td>0.615</td>
</tr>
<tr>
<td>A1113 Simulate Module Architecture</td>
<td>0.017</td>
<td>0.482</td>
</tr>
<tr>
<td>A112 Maintain Component Libraries</td>
<td>0.195</td>
<td>-0.196</td>
</tr>
<tr>
<td>A1131 Create Design</td>
<td>0.608</td>
<td>0.275</td>
</tr>
<tr>
<td>A1132 Capture Schematic</td>
<td>0.219</td>
<td>0.062</td>
</tr>
<tr>
<td>A1133 Simulate &amp; Validate Logic</td>
<td>0.234</td>
<td>0.074</td>
</tr>
<tr>
<td>A1133 Simulate &amp; Validate Timing</td>
<td>0.234</td>
<td>0.074</td>
</tr>
<tr>
<td>A1133 Validate Logic &amp; Timing</td>
<td>0.234</td>
<td>0.074</td>
</tr>
<tr>
<td>A11411 Translate Format</td>
<td>0.123</td>
<td>0.106</td>
</tr>
<tr>
<td>A11412 Schedule Logical Connects</td>
<td>0.060</td>
<td>0.007</td>
</tr>
<tr>
<td>A11413 Assign Design Codes</td>
<td>0.060</td>
<td>0.007</td>
</tr>
<tr>
<td>A11414 Check Logic Design</td>
<td>0.060</td>
<td>0.007</td>
</tr>
<tr>
<td>A11421 Place Parts on Cover</td>
<td>0.259</td>
<td>-0.191</td>
</tr>
<tr>
<td>A11422 Add/Optimize Connects</td>
<td>0.271</td>
<td>-0.229</td>
</tr>
<tr>
<td>A11423 Layout the Etch</td>
<td>0.258</td>
<td>-0.262</td>
</tr>
<tr>
<td>A11431 Run Layout vs. Mfg Rule</td>
<td>0.140</td>
<td>-0.110</td>
</tr>
<tr>
<td>A11432 Run Full Spacing Analysis</td>
<td>0.180</td>
<td>-0.013</td>
</tr>
<tr>
<td>A11433 Add Final Functionality</td>
<td>0.059</td>
<td>-0.002</td>
</tr>
<tr>
<td>A11434 Create Mfg Data Package</td>
<td>0.092</td>
<td>0.038</td>
</tr>
<tr>
<td>A1151 Produce Raw Board Data</td>
<td>0.025</td>
<td>0.030</td>
</tr>
<tr>
<td>A1152 Produce Assembled Board Data</td>
<td>0.025</td>
<td>0.030</td>
</tr>
<tr>
<td>A1153 Create Test Packages</td>
<td>0.100</td>
<td>-0.077</td>
</tr>
<tr>
<td>A12 Judge Productibility</td>
<td>0.143</td>
<td>0.124</td>
</tr>
<tr>
<td>A13 Build &amp; Test Module Prototype</td>
<td>0.088</td>
<td>0.063</td>
</tr>
<tr>
<td>A14 Develop Module Diagnostics</td>
<td>0.099</td>
<td>0.048</td>
</tr>
<tr>
<td>A15 Qualify Module Design</td>
<td>0.102</td>
<td>0.105</td>
</tr>
</tbody>
</table>

**Figure 10. Eigenvectors of Product Design Tasks**

There are no eigenvectors for process design. The eigenvectors for process design are the same as those used for concurrent product-process design. For more details on how the calculations are implemented consult Appendix 4.F.

**4.5.6.3. Interpreting the Eigenstructure**

The eigenstructure of the product design matrix (Figure 10) describes iteration which occurs during product design if the two phases of the design process are not done simultaneously. The first eigenvector of this matrix is primarily composed of circuit design and layout tasks. This mode
indicates that there will be significant iteration between these two portions of the design process.

The second eigenvector is primarily composed of architecture design and circuit design tasks. The mode indicates that there will be iteration between these two phases of the product design process.

4.5.6.4. Total Work Vectors

The total work vectors for design process where the product and the process are designed sequentially are shown in Figure 11. The method for calculating the total work in a two phase sequential design process is described in Appendix 4.G. The first column in Figure 11 gives the total work completed during the product design phase, during which no work is done on the process design. The second column gives the amount of work done during the process design phase, where there is some rework necessary on the product design tasks. The third column sums the first two tasks, showing the total amount of work completed during the two phases.

Comparing the third column of Figure 11 with Figure 9, the total work completed during the concurrent design process, we see that the total amount of work on each task has decreased for every task. In particular, the process design tasks have significantly less work (for example, task A233 Test Manufacturing Process has decreased from 11.34 units of work to 5.81 units of work.) Many of the product design tasks exhibit only slight decreases in the amount of work done (for example, task A11414 Check Logic Design has decreased from 2.00 units of work to 1.97 units of work.)

Implications for the amount of time that it will take to complete the design process are difficult to perceive given the current level of data. We do not have time data for the tasks, so the rework quantities cannot be
turned into time estimates. Although we cannot predict the effect of the
two-phase design process on the time, we suggest that the total amount of
work done doing the design process will decrease, and therefore suggest
that the development costs will also decrease.
<table>
<thead>
<tr>
<th>A1111</th>
<th>Choose Arch. &amp; Technology</th>
<th>2.39</th>
<th>0.31</th>
<th>2.70</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1112</td>
<td>Develop Module Function</td>
<td>2.52</td>
<td>0.32</td>
<td>2.85</td>
</tr>
<tr>
<td>A1113</td>
<td>Simulate Module Architecture</td>
<td>1.50</td>
<td>0.06</td>
<td>1.57</td>
</tr>
<tr>
<td>A112</td>
<td>Maintain Component Libraries</td>
<td>4.05</td>
<td>0.16</td>
<td>4.21</td>
</tr>
<tr>
<td>A1131</td>
<td>Create Design</td>
<td>10.99</td>
<td>0.98</td>
<td>11.97</td>
</tr>
<tr>
<td>A1132</td>
<td>Capture Schematic</td>
<td>4.01</td>
<td>0.23</td>
<td>4.24</td>
</tr>
<tr>
<td>A11331</td>
<td>Simulate &amp; Validate Logic</td>
<td>4.21</td>
<td>0.24</td>
<td>4.45</td>
</tr>
<tr>
<td>A11332</td>
<td>Simulate &amp; Validate Timing</td>
<td>4.21</td>
<td>0.24</td>
<td>4.45</td>
</tr>
<tr>
<td>A11333</td>
<td>Validate Logic &amp; Timing</td>
<td>4.21</td>
<td>0.24</td>
<td>4.45</td>
</tr>
<tr>
<td>A11411</td>
<td>Translate Format</td>
<td>2.64</td>
<td>0.09</td>
<td>2.74</td>
</tr>
<tr>
<td>A11412</td>
<td>Schedule Logical Connects</td>
<td>1.93</td>
<td>0.03</td>
<td>1.97</td>
</tr>
<tr>
<td>A11413</td>
<td>Assign Design Codes</td>
<td>1.93</td>
<td>0.03</td>
<td>1.97</td>
</tr>
<tr>
<td>A11414</td>
<td>Check Logic Design</td>
<td>1.93</td>
<td>0.03</td>
<td>1.97</td>
</tr>
<tr>
<td>A11421</td>
<td>Place Parts on Cover</td>
<td>6.13</td>
<td>0.41</td>
<td>6.53</td>
</tr>
<tr>
<td>A11422</td>
<td>Add/Optimize Connects</td>
<td>5.79</td>
<td>0.21</td>
<td>6.00</td>
</tr>
<tr>
<td>A11423</td>
<td>Layout the Etch</td>
<td>5.53</td>
<td>0.17</td>
<td>5.70</td>
</tr>
<tr>
<td>A11431</td>
<td>Run Layout vs. Mfg Rule</td>
<td>3.51</td>
<td>0.07</td>
<td>3.57</td>
</tr>
<tr>
<td>A11432</td>
<td>Run Full Spacing Analysis</td>
<td>4.25</td>
<td>0.32</td>
<td>4.57</td>
</tr>
<tr>
<td>A11433</td>
<td>Add Final Functionality</td>
<td>2.12</td>
<td>0.03</td>
<td>2.15</td>
</tr>
<tr>
<td>A11434</td>
<td>Create Mfg Data Package</td>
<td>2.81</td>
<td>0.04</td>
<td>2.85</td>
</tr>
<tr>
<td>A1151</td>
<td>Produce Raw Board Data</td>
<td>1.56</td>
<td>0.01</td>
<td>1.57</td>
</tr>
<tr>
<td>A1152</td>
<td>Produce Asmbld Board Data</td>
<td>1.56</td>
<td>0.01</td>
<td>1.57</td>
</tr>
<tr>
<td>A1153</td>
<td>Create Test Packages</td>
<td>3.04</td>
<td>0.05</td>
<td>3.09</td>
</tr>
<tr>
<td>A12</td>
<td>Judge Productibility</td>
<td>3.71</td>
<td>1.25</td>
<td>4.96</td>
</tr>
<tr>
<td>A13</td>
<td>Build &amp; Test Module Prototype</td>
<td>2.82</td>
<td>1.23</td>
<td>4.05</td>
</tr>
<tr>
<td>A14</td>
<td>Develop Module Diagnostics</td>
<td>3.10</td>
<td>0.29</td>
<td>3.39</td>
</tr>
<tr>
<td>A15</td>
<td>Quality Module Design</td>
<td>3.23</td>
<td>0.29</td>
<td>3.52</td>
</tr>
<tr>
<td>A211</td>
<td>Develop Overall Scheme</td>
<td>2.40</td>
<td></td>
<td>2.40</td>
</tr>
<tr>
<td>A2121</td>
<td>Design Details of Workstations</td>
<td>2.03</td>
<td></td>
<td>2.03</td>
</tr>
<tr>
<td>A2122</td>
<td>Build Workstation Prototype</td>
<td>1.41</td>
<td></td>
<td>1.41</td>
</tr>
<tr>
<td>A2123</td>
<td>Dvlp Tool Generation Capabil</td>
<td>1.41</td>
<td></td>
<td>1.41</td>
</tr>
<tr>
<td>A2124</td>
<td>Test Workstation Prototype</td>
<td>1.97</td>
<td></td>
<td>1.97</td>
</tr>
<tr>
<td>A213</td>
<td>Develop Process Tests</td>
<td>2.15</td>
<td></td>
<td>2.15</td>
</tr>
<tr>
<td>A221</td>
<td>Construct and Lay Out Facility</td>
<td>2.37</td>
<td></td>
<td>2.37</td>
</tr>
<tr>
<td>A222</td>
<td>Install Workstations</td>
<td>2.85</td>
<td></td>
<td>2.85</td>
</tr>
<tr>
<td>A223</td>
<td>Train Operators</td>
<td>2.37</td>
<td></td>
<td>2.37</td>
</tr>
<tr>
<td>A2311</td>
<td>Develop Board Processes</td>
<td>3.13</td>
<td></td>
<td>3.13</td>
</tr>
<tr>
<td>A2312</td>
<td>Develop Assembly Processes</td>
<td>3.10</td>
<td></td>
<td>3.10</td>
</tr>
<tr>
<td>A2313</td>
<td>Develop Assy Mfg Tests</td>
<td>2.72</td>
<td></td>
<td>2.72</td>
</tr>
<tr>
<td>A2321</td>
<td>Gerate Board Softools</td>
<td>2.52</td>
<td></td>
<td>2.52</td>
</tr>
<tr>
<td>A2322</td>
<td>Make Board Fixtures</td>
<td>2.52</td>
<td></td>
<td>2.52</td>
</tr>
<tr>
<td>A2323</td>
<td>Generate Assembly Softools</td>
<td>3.42</td>
<td></td>
<td>3.42</td>
</tr>
<tr>
<td>A2324</td>
<td>Make Assembly Fixtures</td>
<td>2.51</td>
<td></td>
<td>2.51</td>
</tr>
<tr>
<td>A2325</td>
<td>Generate Assy Test Pgms</td>
<td>2.71</td>
<td></td>
<td>2.71</td>
</tr>
<tr>
<td>A233</td>
<td>Test Mfg Process</td>
<td>5.81</td>
<td></td>
<td>5.81</td>
</tr>
</tbody>
</table>

**Figure 11. Total Work Vectors for Sequential Design of Product and Process**
4.5.7. Iteration in Practice

In actual practice, module design is highly iterative. The sources of iteration were documented by interviews with Digital engineers who were familiar with module design, but not those who provided the data used in the construction of the matrix. This is intended to be an independent confirmation that the work transformation model of module design accurately captures the behavior of the design organization. The engineers described the greatest source of iteration as occurring within layout and simulation.

The description given by the engineers is confirmed by a note in the SADT report. As stated in the report: "Design and Validate Logic is composed of three subprocesses: Create Design, Capture Schematic, and Simulate Logic and Timing. The process is highly iterative with the Create Design subprocess being highly dependent on feedback from the other two subprocesses." That is, there is much iteration among the tasks which create the design, create the details of the layout (schematic), and simulating the performance.

This iteration occurs both as errors in the design are diagnosed and eliminated, and as the performance of the system is improved and optimized. The complexity of the system makes it impossible to create an error-free design, and it is also impossible to anticipate or predict the performance of the system without running extensive simulations.

In practice there is, however, little iteration between process design and product design. Process design issues are expressed to the product designers as specifications and design rules. If product design were done concurrently with product design, the product design cycle would be slowed significantly. When a product is designed which does not meet with the
desires of the process engineers, they are faced with the problem of producing it despite their concerns. It may be produced for a higher cost than desired, or have a lower yield, but it will still be possible to produce it. The decision that the organization must make is whether to delay the release of the product in order to lower its cost. Because of the tight lead time pressures which exist in the workstation industry, this choice is regularly taken to shorten the lead time at the expense of cost. In other products and other industries we would expect to see a different approach to this choice.

4.5.8 How to Use the Work Transformation Model to Improve the Module Design Process

The work transformation model has been shown to be accurate in predicting the design features which require the greatest amount of iteration. The intent of this modeling effort is to be able to apply these models to improve the design process.

One criticism of these models is that they are somewhat tautological, that is, that the tasks which they identify to be highly iterative were already known by the organizations studied to be characterized by iteration. One of the efforts in studying module design at Digital Equipment was to keep the modeling effort sufficiently separate from the design organization so that the identification of the typical design problems would not be tainted by familiarity. The goal was to verify that the model would be able to identify the heart of the problem.

The identification of the design modes can be used to justify why tasks might be done sequentially or concurrently. It is not efficient to do the assembly process design tasks in parallel with detailed product design, because there would be too much iteration. Instead, the product design is
specified, and then the process design task is undertaken. The organization may or may not have good measures of this efficiency. The matrix analysis comparing the concurrent design process, section 4.5.5, with the sequential design process, section 4.5.6, can potentially provide such a measure.

4.5.9. Conclusion

A Work Transformation Matrix of the module design process has been constructed. Analysis of the matrix shows that there are several distinct modes of iteration, some of which include both product and process design tasks, and some of which are focused on just product or process design. In practice, iteration does not occur between product and process design because of strong time pressures. The main iteration which occurs is between circuit design and layout, which is the main design mode in the product design portion of the matrix. This type of analysis might serve as a tool to indicate how separating the product and process design into different phases might affect the time it takes to complete the design project. The sequential form of analysis of the Work Transformation Model bridges the gap between the parallel WTM and the Sequential Iteration Model of Chapter 3. This combination has the advantages of the superior realism of the Work Transformation Model with the power of the decision making tool of the Sequential Iteration Model.

4.6. Experimental Verification

This section compares two alternative design strategies for the Delta Design Game, an engineering design exercise. We first analyze these strategies using the Work Transformation Matrix and show that one of the strategies is expected to display a faster solution time. We then
demonstrate experimentally the difference in development time by observing eight design teams working on the problem using the two strategies. We found that the "decoupling strategy" suggested by the model reduced solution time while maintaining quality of the technical solutions.

Experiments have been used in other settings to test hypotheses about which factors affect the ability of designers to accomplish their task [Jakiela and Orlikowski 1990, Papalambros 1988]. Experimental settings increase the ability of the researcher to control the design environment and to gather many data rapidly.

4.6.1. Delta Design Game

The Delta Design Game is an exercise originally used to demonstrate to engineering students that design is a process of negotiation among several conflicting disciplines and requirements [Bucciarelli 1990]. The games has been adopted in order to demonstrate how two different design strategies for the same design problem can be judged using the Work Transformation Matrix.

In the Delta Design Game, four designers are responsible for designing a residential structure in a two dimensional world. (See Figure 12 for an example design; more details about the game can be found in Appendix 4.H.) The building materials are triangular elements (deltas) which must be combined subject to a set of design specifications (thermal, structural, aesthetic and fiscal). Deltas come in both red (heat source) and blue (heat sink) varieties. Each of the participants is assigned a role in the design process, and associated with each role is some specific technical knowledge about the specifications. The roles are Thermal Engineer, Structural Engineer, Architect, and Project Manager.
Design 4

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Quality pts.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Design 4</td>
</tr>
<tr>
<td>2</td>
<td>Time (min)</td>
</tr>
<tr>
<td>3</td>
<td>Total Quality</td>
</tr>
<tr>
<td>4</td>
<td>Cost</td>
</tr>
<tr>
<td>5</td>
<td>Area (Deltas)</td>
</tr>
<tr>
<td>6</td>
<td>% blueness</td>
</tr>
<tr>
<td>7</td>
<td>Re and Rb</td>
</tr>
<tr>
<td>8</td>
<td>Load</td>
</tr>
<tr>
<td>9</td>
<td>Overlap safety</td>
</tr>
<tr>
<td>10</td>
<td>Avg temp</td>
</tr>
<tr>
<td>11</td>
<td>Max temp</td>
</tr>
<tr>
<td>12</td>
<td>Min temp</td>
</tr>
<tr>
<td>13</td>
<td>Aesthetics</td>
</tr>
</tbody>
</table>

**Figure 12. Example Design**

The game takes about two hours to complete, including thirty minutes to train the participants in their various roles. During the training period the participants do not interact so that their areas of expertise remain distinct.

**4.6.1.1. Alternative Design Strategies**

The standard design strategy, as described in the original instructions to the game, is as follows. The architect suggests a structure by arranging the deltas in a proposed design layout. The three other participants analyze the aspects of structure under their domain and suggest design improvements. A new layout is proposed and the analysis is repeated. This iteration process continues until the design takes its final form (meets the target specifications). This process is shown in Figure 13.
Figure 13. Standard Design Strategy

The standard strategy (also referred to as the baseline strategy) can be represented by a Work Transformation Matrix as follows. (See Figure 5.) The first five tasks in the matrix are the design tasks (where the participants lay out a suggested design). The other ten tasks are the analysis tasks, where the participants judge the design against the given criteria. There are no direct information flows from one design task to another, nor is there a direct dependency from one analysis task to another. Nevertheless, the matrix is fully coupled. The strengths of the dependencies were determined based on the author's participatory and observational experience with the design game.
Figure 14. Matrix for Standard Strategy

Analyzing the eigenvalues and eigenvectors of the above matrix identifies the issues driving the design iteration. The eigenvectors corresponding to the largest eigenvalue (primary design mode) is primarily composed of tasks A, B, F, G, H, and M. (See Figure 15.) These tasks are associated with the thermal design problem and the cost of construction. The eigenvector corresponding to the second largest eigenvalue (secondary design mode) is primarily composed of tasks D, L, and O. These tasks are associated with the structural design problem (loads and moments.) We see that these two design modes are somewhat independent (i.e. they have different tasks which are heavily weighted.)
Figure 15. Dominant Eigenvectors of Standard Strategy

One method of working on a complex problem where there is some independence is to split the problem into smaller pieces, and to work on the pieces separately [Alexander 1964, Suh 1990]. We identified a design strategy which would allow the designers to separate the two design modes. The cost and thermal design criteria scale reasonably well from a small portion of the structure up to the entire building. The recognition of independence of thermal and cost criteria to scale is not apparent from the structure of the matrix, but relies on other knowledge about the problem being solved.
Figure 16. Building Block Strategy

This idea can serve as the basis for a design strategy (See Figure 16). During the first phase of the design process, the design team would try to find a building block (group of 2-6 deltas) which meets the local thermal criteria, and seems likely to produce a lower cost structure. This building block would then be replicated, and the blocks joined into an overall structure. The complete structure would then be checked against the structural criteria. If the building fails to meet any of the criteria, either the building blocks would be joined in a new configuration, or the building block itself would have to be redesigned. This process would be repeated until the structure was complete.
Figure 17. Matrix for Building Block Strategy

The building block strategy was also represented as a Work Transformation Matrix (See Figure 17). The building block matrix is similar to the matrix for the original strategy, although a few of the tasks have to be redefined and resequenced. The matrix is still fully coupled.

Looking at the dominant eigenvectors of this matrix, we see that the two primary design modes for this matrix are independent, as we expected when we selected this design strategy (See Figure 18). The tasks which are heavily weighted within each of the two dominant eigenvectors contain non-overlapping heavily-weighted tasks.
Figure 18. Dominant Eigenvectors for Building Block Strategy

Since the eigenvalues are most closely associated with the number of iterations it takes to find the design solution, more information about task times would be necessary to estimate the relative amount of time that it takes to complete the two design processes. We did not attempt to identify the task times. We believed that the building block strategy would dominate the original strategy because it takes a shorter amount of time per iteration, not because it takes fewer iterations. Comparing the tasks, we would expect it to take less time to choose interfaces which meet the thermal constraints when there are 2-6 deltas than for the complete matrix.

4.6.1.2. Experimental Method

In implementing the experiment, we wish to test to see if there is a significant difference in design time between groups who use the two strategies.
We formed eight groups of four undergraduate engineering students. Each member of each group was instructed in the overall nature of the design problem, their area of expertise, and the design strategy they were to employ.

We observed each group performing the design task and we recorded the total time taken until a final design was produced. We then recorded the final design in order to calculate a quality score. This was needed to see if there were any differences in the quality of the resulting designs.

The quality score codifies our attempt to unify all of the various design constraints and criteria into one measure. The experimental groups did not use the quality score to evaluate the designs. They designed to the original design criteria. The quality score evaluation was done subsequently, for our own purposes.

The quality score penalized groups for those criteria which they failed to fulfill, while rewarding groups which exceeded design guidelines. For several of the constraints, there were two levels of constraint. These levels would be expressed as a desired cost goal, and also a level with a 10% higher allowance. On these types of constraint the penalty was not large unless the group failed to meet the relaxed condition, for which they were penalized heavily.

4.6.1.3. Experimental Results

The graph below (Figure 19) shows the relative times and quality scores for the eight design groups. We observe that the Building Block design teams took significantly less time than the Baseline design teams, while there is no significant difference in quality between the approaches. The significance of the differences between the mean times was tested using a two-sample t-test [DeGroot 1986]. The difference between the means
of the times for the standard strategy (84.25) and the building block strategy (55.25) is statistically significant at a 0.005 level. The difference between the mean of the quality score for the standard strategy (9.30) and the building block strategy (9.64) is not significant, even at a 0.20 level of significance.

(The one datum which is slower and of lower quality than the other data is explained as follows. The groups were told it would take 60-90 minutes to complete the design. They knew after 105 minutes that they were taking more time than allotted, although their design was improving. At this point they stopped working on the problem. It is our hypothesis that, had they continued working, this group's quality score would have come more in line with the other groups', at the expense of still more time.)

![Figure 19. Time vs. Quality](image)
There are no other significant differences in the design results between the two design approaches. Complete experimental data are presented in a previous thesis [Gopal 1992].

4.6.2. Discussion

The building block approach does not fully decouple the design problem; there is still information transfer required between the two design sub-problems. The most important feature of choosing the building block (first phase) which affects the structural design (second phase) is the setting of sufficient overlap between the deltas, so that the internal moment design criterion can be more easily fulfilled.

By suggesting a building block design strategy, we may have given more information to the design groups, explicitly or implicitly. The goal of this experiment was not to keep the level of knowledge constant, rather to see if the matrix method is able to explain any differences between performance of group which use alternative strategies. What is important is that the two approaches are significantly different, and that the differences described in the matrix account for a significant portion of the difference.

We are not able to suggest design strategies directly from observation of the mathematics of the matrix. Rather, knowledge of the design environment is required. This is not necessarily bad; we would not expect design structure analysis to be able to improve the design without domain-specific knowledge. The observation of the independence of the main design modes can, however, suggest to the designer where to look for strategies which will separate the modes into smaller problems.

We are led to wonder under what conditions decoupling is appropriate. In the delta design problem we were able to identify a
decoupling strategy because of the independence of the design modes. We hypothesize that the independence of the primary design modes indicates that a decoupling strategy is technologically feasible.

Not all decoupling strategies will necessarily improve the design process. If the problem is divided across one or more key technical issues, then coordination between the two parts of the design process becomes an important and difficult task. It may be necessary to redefine some of the design tasks to facilitate decoupling. Also, it is often useful to look for opportunities to use scaling in order to decouple a design problem.

The industrial design environments discussed earlier (brake system design in section 4.4 and electronics module design in section 4.5) have design matrices which exhibit some degree of independence among the main design modes. The organizations did not exploit this independence specifically in order to solve their particular design problem. Both of these studies were conducted at firms which have extensive experience with their technical problems, and they feel that they have a good grasp of the technical issues which control their design problem. The goal of this modeling is to help identify the important features which control design iteration, which can help improve the management of design projects.

4.6.3. Conclusion

We have been developing models of the design process in order to increase our understanding of design and to provide managers with tools that they can use to improve their control of such projects. This section tests the validity of the Work Transformation Model by comparing the predictions of the model with observation of design in an experimental setting. The model suggests that the building block design strategy exploits independence in the technical issues. Also, the reduced magnitude of the
design problem within a building block enables faster iterations. The experimental data support the superiority of the building block design strategy. The work transformation matrix model can also help to identify where a design strategy exhibiting such independence is possible or likely. Our future work will attempt to make the identification of superior strategies more structured. We also hope to test the ability of the design model to predict the success of a design strategy in an industrial design setting.

Appendix 4.A. Doing a Portion of the Work in Each Iteration

This appendix contains an extension to the Work Transformation Matrix model. This extension add generality to the original model, but are only slight modifications. The primary insight obtained from the analysis is that the eigenvalues and eigenvectors of $A$ are still the most important analytical features, even with a more general model.

In the original model all of the work is executed during every iteration stage. We wish to relax this assumption, allowing a specified fraction of the work to be done in each iteration. During each iteration stage, the proposed policy is to do a proportion $\rho$ of all work on every task in each stage. The work which is not completed during the current iteration stage remains to be completed in future stages. Work which is completed creates work for other tasks as in the original model. The relation between successive work vectors is therefore:

$$u_{t+1} = [(1-\rho)I + \rho A]u_t \quad 0<\rho\leq1$$

We define a modified work transformation matrix $A^*$ such that:

$$A^* = [(1-\rho)I + \rho A]$$
so that:

\[ u_{t+1} = A^* u_t \]

We can find the eigenvectors and eigenvalues of the matrix \( A^* \):

\[ A^* = \left[ (1-\rho)I + \rho \Lambda S S^{-1} \right] \]

\[ A^* = S \left[(1-\rho)I + \rho \Lambda \right] S^{-1} \]

The matrix \([(1-\rho)I + \rho \Lambda] \) must be the eigenvalue matrix of \( A^* \) since it is diagonal and it is similar to \( A \). It is seen that the eigenvector matrix \( S \) of \( A^* \) is the same as that of \( A \). The eigenvalue matrix of \( A^* \) is a convex combination of \( \Lambda \) and \( I \). Since the eigenvalues have been increased, the convergence has been slowed (which is to be expected since we are only doing a proportion of the work in each stage.) The shapes of the design modes which described convergence remain unchanged.

**Appendix 4.B. Time-Based Interpretation of the Work Transformation Matrix**

This appendix develops an explicit time-based interpretation of the work transformation matrix. It is shown below that this interpretation is, in fact, identical to the original way in which time was considered. The basis for the new formulation uses the vector \( u^\dagger \) as a work time vector:

\[ u^\dagger_{t+1} = A^\dagger u^\dagger_t \]

The initial work time vector is the initial work vector weighted by the time for each task:

\[ u^\dagger_0 = W u_0 \]

where \( W \) is a diagonal matrix of the task times \( w_i \). This vector indicates that the full amount of time remains to be completed on each task.
Each element in the work time transformation matrix $A^\dagger$ is the amount of work time that one hour of task $j$ creates for task $i$, or:

$$a^\dagger_{ij} = \frac{w_i}{w_j} a_{ij}$$

The work time transformation matrix can be written compactly as:

$$A^\dagger = WAW^{-1}$$

(The matrix $W$ is always invertible. It is a diagonal matrix with non-zero diagonal entries.)

Repeating the analysis done for the original system, the total work time vector can be found by summing the work time vectors:

$$U^\dagger = \sum_{t=0}^{T} u^\dagger_{t+1}$$

which can be written as:

$$U^\dagger = WS \left[ \sum_{t=0}^{T} \Lambda^t \right] S^{-1}W^{-1}u^\dagger_0$$

Substituting for the initial work time vector $u^\dagger_0$:

$$U^\dagger = WS \left[ \sum_{t=0}^{T} \Lambda^t \right] S^{-1}W^{-1}Wu_0$$

or:

$$U^\dagger = WS \left[ \sum_{t=0}^{T} \Lambda^t \right] S^{-1}u_0$$

which reduces to:

$$U^\dagger = WU$$

which is the expression originally given for weighting the total work vector by the task times.
Appendix 4.C. The Jordan Form

This appendix presents the Jordan form, which is a more general form of the eigenvalue/eigenvector decomposition of square matrices. (For more detail on the Jordan form see [Strang 1980].) Every matrix $A$ with $s$ linearly independent eigenvectors can be decomposed into:

$$A = MJM^{-1}$$

where $J$ is a Jordan matrix and matrix $M$ is invertible. A Jordan matrix is one which has the following structure:

$$J = \begin{bmatrix} J_1 \\ \vdots \\ J_s \end{bmatrix}$$

where the blocks have the form:

$$J_i = \begin{bmatrix} \lambda_i & 1 \\ & \ddots \\ & & \lambda_i \end{bmatrix}$$

Note that the eigenvalue/eigenvector decomposition is a special case of the Jordan form where each of the blocks are 1x1.

The Jordan form is used when there are repeated eigenvectors for which linearly independent eigenvectors do not exist. For each such block the first column of $M$ contains the eigenvector corresponding to $\lambda$, and the subsequent columns contain other quantities. When interpreting such a matrix for dynamic analysis, only the eigenvectors need be considered.

The total work time vector is calculated similarly to the derivation described in section 4.2:

$$U = M \left( \sum_{t=0}^{T} J^t \right) M^{-1} u_0$$
although for such a matrix it is not as straightforward to calculate:

\[
\sum_{t=0}^{T} J^t
\]

as it is with the eigenvalue matrix.

The situation of a non-diagonalizable matrix is included for completeness. The need for these mathematics has not arisen in our experience with work transformation matrices.

(I would present such a matrix here, but I have not found a nonnegative coupled matrix where the eigenvector matrix is not invertible.)

Appendix 4.D. The Perron-Frobenius Theorem

This appendix states the Perron-Frobenius Theorem. More information, including the proof of the theorem, can be found a book on matrix theory [Marcus and Minc 1964].

The Perron-Frobenius Theorem:

Let A be a nxn coupled nonnegative matrix. Then:

(i) A has a real positive eigenvalue \( r \) (the maximal eigenvalue of A) which is a simple root of the characteristic equation of A. If \( \lambda_i \) is an eigenvalue of A, then:

\[
|\lambda| \leq r.
\]

(ii) There exists a positive eigenvector corresponding to \( r \).

(iii) If A has \( h \) characteristic roots of modulus \( r \): \( \lambda_0 = r, \lambda_1, \cdots, \lambda_{n-1} \) then these are the \( h \) distinct roots of:

\[
\lambda^h - r^h = 0;
\]

\( h \) is called the index of imprimitivity of A.
(iv) If \( \lambda_0, \lambda_1, \ldots, \lambda_n \) are all the eigenvalues of \( A \) and \( \theta = \exp(i2\pi/h) \), then 
\( \lambda_0 \theta, \lambda_1 \theta, \ldots, \lambda_n \theta \) are \( \lambda_0, \lambda_1, \ldots, \lambda_n \) in some order.
(v) If \( h > 1 \), then there exists a permutation matrix \( P \) such that:

\[
\begin{bmatrix}
0 & A_{12} & 0 & \cdots & 0 & 0 \\
0 & 0 & A_{23} & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & \cdots & 0 & A_{h-1,h} \\
A_{h,1} & 0 & 0 & \cdots & 0 & 0 \\
\end{bmatrix}
\]

where the zero blocks down the main diagonal are square.
(vi) A cannot have two linearly independent positive eigenvectors.

The theorem is self explanatory except for part (iv). Part (iv) says that 
if the maximal eigenvalue is periodic (there is more than one eigenvalue of 
magnitude \( r \)), then all eigenvalues will be periodic.

**Appendix 4.E. Sensitivity of the Eigenvectors to Weights**

This appendix describes how we have verified that the brake system 
eigenstructure is robust against small changes in the weights chosen in 
the Work Transformation Matrix.

There are two parts to this robustness, first to demonstrate how 
scaling the Work Transformation Matrix does not affect the eigenstructure; 
second to demonstrate that reasonable choices for the weights in the matrix 
will lead to the same sets of critical parameters in the design modes.

First we look at the scaling of the matrix \( A \). As the matrix was 
defined in section 4.4, there are three weights, low, medium and high. For 
any numerical choices of these three parameters, the quantities which 
affect the major tasks in the design modes are the ratios between the 
weights, not the values of the weights themselves. In the equation:
\[ kA = S(k\Lambda)S^{-1} \]

we see that when the matrix \( A \) is scaled by a constant factor, the
eigenvector matrix does not change, nor do the relative magnitudes of the
eigenvalues. Our interpretation of which the important tasks are will
therefore not change with scaling.

We therefore want to look at the effect of relative scaling, rather than
absolute scaling. We will define two quantities which characterize the
relative scaling. First is \( \alpha \), the ratio of the magnitude of the medium
dependency relative to the low dependency; the second is \( \beta \), the ratio of the
magnitude of the large dependency relative to the medium dependency.

Both of these quantities, \( \alpha \) and \( \beta \), should not be less than 1 (since this
would mean that the medium had a smaller weight than the low, or the
heavy had a smaller weight than the medium). We would also not expect to
see realistic values exceed some maximum quantity, which would indicate
that the medium level importance was very much more important than the
light importance (likewise with large and medium importance.) For the
purposes of this discussion the values of \( \alpha \) and \( \beta \) have been capped at 5.

Twenty-six different plausible weighting schemes which cover the
ranges \( 1<\alpha<5 \) and \( 1<\beta<5 \) were evaluated to see the two main design modes.
The eigenvectors were normalized to unit length and the tasks which had a
weighting larger than 0.16 in the eigenvectors were identified. The sets of
tasks for the two eigenvectors under each weighting scheme were
evaluated.

For the dominant eigenvector (stopping distance) there are eight
tasks which appear as important in most of the primary eigenvectors.
These tasks are Pressure at Rear Wheel Lockup, Brake Torque vs.
Skidpoint, Vehicle Deceleration Rate, Dash Deflection, Pedal Force
(Required), Pedal Mechanical Advantage, Lining Material-Front, and
Booster Reaction Ratio. These tasks are considered the nominally
important tasks.

The second eigenvector, or the eigenvector which corresponds to the
second largest positive eigenvalue, has even less variation. This
eigenvector is associated with the overheating problem. There are six tasks
which were identified to appear in most of the second eigenvectors. These
tasks are Splash Shield Geometry-Front, Air Flow under Car/Wheel Space,
Temperature at Components, Rotor Cooling Coefficient, Rotor Width, and
Rotor Material. The are the nominal tasks in the second eigenvector.

We then evaluated which of the eigenvectors for all of the weighting
schemes were not equivalent to the nominal eigenvectors. Each tasks
which was identified as having a weighting larger than 0.16 which was not
one of the tasks in the nominal eigenvectors, or one of the tasks in the
nominal eigenvectors which had a weighting smaller than 0.16 is
considered a misfit, in that the identified eigenvector does not have the
same important tasks as the nominal eigenvector. The number of misfits
for each of the weighting schemes was totaled. Figure 20 show the number
of misfits for each of the 26 weighting schemes chosen as a function of α
and β.
Figure 20. Misfits with Nominal Eigenvectors vs. $\alpha$ and $\beta$

We see that there is a large region of the space where there are no misfits between the nominal eigenvectors and the identified eigenvectors (the area enclosed within the polygon). Also, a number of the other weighting schemes had only 1 or 2 misfits. As long as $\alpha$ and $\beta$ were chosen within this range, we could have expected that the eigenvectors would have appropriately identified the stopping distance and the overheating problems as the primary sources of iteration.

This appendix demonstrates that many reasonable choices for $\alpha$ and $\beta$ which give similar eigenvectors for the brake system problem. In general, it is necessary to check that the weightings produce a valid and robust set of eigenvectors for any given problem.

Appendix 4.F. Sequential Phases of Work Transformation Matrix

This appendix describes how a Work Transformation Matrix can be interpreted to describe a design process where the iteration is completed in
multiple phases. During each phase, work is completed on tasks which fall within the current phase, and rework is done on tasks which fall in previous phases.

(For the purposes of this example, the two phases are product design and process design, which corresponding to the module design problem described in section 4.5.)

First, the work transformation matrix $A$ is divided into blocks. The upper left block contains sections which describe iteration internal to the product design tasks. The lower right block describes iteration internal to the process design tasks. The lower left block describes inputs from the product design tasks to the process design tasks. The upper right block describes feedbacks from the process design tasks to the product design tasks. These blocks can be better seen in the following equation:

$$A = A^{\text{product} \& \text{process}} = \begin{bmatrix} A^{\text{product}} & \text{feedbacks} \\ \text{inputs} & A^{\text{process}} \end{bmatrix}$$

The iteration process has been divided into two phases. First is the product design phase, where the product design tasks are iterated until they come to completion. No work is done on the process design tasks during the product design phase.

Second is the process design phase, where the completed product design is taken as an input and the process design tasks are iterated until they come to completion. During the process design phase some work will need to be done on the product design phase due to the feedbacks in matrix $A$.

The calculation of the total work vector $U$ is described by the following equation:
\[
U = \begin{bmatrix}
\sum_{t=0}^{T} u_{t}^{\text{product}} \\
0 \\
\vdots \\
0
\end{bmatrix} + \sum_{t=0}^{T} u_{t}^{\text{process}}
\]

where the first term in the equation is the work done during the product design phase, and the second term is the work done during the process design phase.

The work vectors in the product design phase are calculated just as the work vectors in the original parallel model described in section 4.2, although using the restricted upper left block of the matrix A:

\[
u_{t}^{\text{product}} = [A^{\text{product}}]^t u_{0}^{\text{product}}
\]

where:

\[
u_{0}^{\text{product}} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}
\]

Therefore, the eigenstructure of the upper left block of A will be used to describe iteration during the product design phase, as described in subsection 4.2.3 of the text.

However, the work vectors in the process design phase are calculated somewhat differently. Since some work is done on the product design tasks during the process design phase, we must use the entire matrix A to calculate the work vectors:

\[
u_{t}^{\text{process}} = [A] u_{0}^{\text{process}}
\]

The initial work vector has a different type of construction. Initially, no work needs to be done on the process design tasks, although all work
remains to be done on the process design tasks. These requirements are reflected in the initial work vector:

\[
\mathbf{u}_0^{\text{process}} = \begin{bmatrix}
0 \\
\vdots \\
0 \\
1 \\
1
\end{bmatrix}
\]

The eigenvalues and eigenvectors of the complete matrix \( A \) will describe iteration during this phase, although the weights obtained:

\[ S^{-1}\mathbf{u}_0^{\text{process}} \]

(and therefore the total amount of work completed) will be different from:

\[ S^{-1}\mathbf{u}_0 \]

the weights used in the parallel design process.

For an example of sequential work transformation matrix application please see section 4.5.6 of the text.

**Appendix 4.G. Details on Delta Design Exercise**

This appendix gives more detail about the design exercise. The exercise involves a group of four people working together on a design task. The object of the design is to construct a two dimensional structure made out of triangular red and blue elements (deltas).

Each of the four designers has one of the following roles: project manager, architect, structural engineer, and thermal engineer. The project manager is responsible for meeting cost targets, the architect is responsible for aesthetic considerations, the thermal engineer for temperature constraints, and the structural engineer for meeting the specified loads and moments.
The project manager is responsible for costs. Each element in the structure has a cost, and there is a cost for joining elements together. The cost functions are nonlinear.

The architect is responsible for aesthetic concerns. The goal of the structure is to produce a smooth exterior with a jagged interior (although these goals are not specified fully.) Also, it is desirable to use no more than 60% blue deltas.

The thermal engineer is responsible for thermal specifications. The temperature is a function of how many heat generating elements there are and how much radiating (exterior) surface exists. There are both local and overall constraints (maximum and minimum) on temperature.

The structural engineer is responsible for setting the points of attachment and checking the loads and moments. Each point of attachment can support the weight of than 20 deltas, although it is desirable to have a comfortable safety margin. Also, the joints in the structure must be capable of carrying internal moments, which occur due to cantilevering.

We have simplified the exercise slightly from its original version [Bucciarelli and Goldschmidt 1989]. We have removed the effects of variable gravity which are discussed, and we have simplified a few of the formulas. These changes have improved the clarity of the goals of the exercise to the participants, without making it a trivial design exercise. More details about the game, including specific functional forms of the constraints, can be found in a previous thesis [Gopal 1992].
Chapter 5. Discussion of Modeling Philosophy

Models are useful tools for studying idealized processes, but are limited in their application by their assumptions. Models are necessarily less complex than the real world; this is what gives them power. It is important to determine for any model whether its behavior includes the important behavior in the real world situation without being so simple as to lose relevance. This chapter discusses to what extent the models described in Chapters 3 and 4 fulfill the requirements of correctly describing the behavior of an engineering design organization, and what methods can be used to test to see if the behavior is correctly modeled.

5.1. Model Validity

There are several ways that a mathematical model can be validated. These include

(1) do model assumptions make sense?
(2) do model predictions make sense?
(3) does the behavior of the model match measured behavior?
(4) is this model better than other available models?
(5) does the model work in a reduced complexity (experimental) setting?
(6) can the model be used to control successfully?

I will discuss each of these verification issues in greater detail, and compare to see how the models discussed in the thesis meet each of the criteria.

(1) First, the assumptions of the model must represent the real situation being modeled. A mathematical model is only as strong as its assumptions. Also, the assumptions must not be so simplistic as to eliminate the complexity of the system being studied.
The probabilistic sequential iteration model creates a restrictive set of assumptions. The idea of rework's being a probabilistic choice is not an accurate description of iteration in all situations.

The work transformation model also uses a restrictive set of assumptions, although there is more correspondence between these assumptions and a realistic description of design iteration. An assumption of parallelism of coupled design tasks is often true in an organization, although the fully parallel iteration of the work transformation model, where all tasks are done during each iteration, is an extreme which is not necessarily accurate. It is also true that iterations take less time in the second and subsequent stages. The specific assumption that the amount of time per iteration is a linear function of the previous iteration is not likely to be strictly true, but can serve as a good approximation for the decreasing amount of work for each iteration.

Second, we must see how the predictions of the model match with our intuition about the system being modeled. Predictions which do not match our perceptions are both useful and problematic. Non-intuitive predictions can be an indication that the model does not correctly capture the modeled behavior, and therefore an indication that the model is incorrect. Alternatively, non-intuitive predictions can show an area where the model is able to describe a behavior which is too complex to be understood without the use of the model. Therefore, we must find and examine non-intuitive results in order to characterize the power and limitations of the model.

The results of the probabilistic sequential iteration model suggest that if the expected time and the variance of the project time are to be lessened, the longer tasks should not be attempted until later in the project, and also that tasks should not be attempted until after their important
predecessors are completed. Both of these suggestions make intuitive sense. The predictions of this model therefore meet the reasonability criterion.

(2) The predictions of the work transformation model are an indication of which groups of tasks will require the greatest amount of iteration during the design process. For both the brake system design example and the module design example the groups of tasks which the model indicates will require significant iteration are those which require iteration during the actual design process. This model also therefore meets the criterion of having the predictions be intuitively satisfying.

(3) The test of having the quantitative results of the model match measured behavior is more rigorous. This criterion requires that the quantitative results be substantially similar to observed values. Neither of the models in this thesis is able to meet this criterion because quantitative data were not gathered. We have not been able to compare, for example, the total number of iterations in the model with total number of observed iterations.

(4) A further criterion for a model is to see if the model applied is better than all other models in the current system. This criterion assumes that a model is necessary and suggests that the best available model is good enough for the purposes of analysis. This test is well met by the models in the thesis, strictly because there are no other explicit quantitative models of design iteration available.

(5) The next criterion is the appropriateness of the model in a reduced complexity or experimental setting. An experiment is constructed in order to reduce the amount of variability in a system being studied, and to increase the repeatability. These restrictions ensure that the data gathered
are accurate, but it is possible (in a system as complex as an engineering design organization) that an experiment will reduce the complexity so that the system is no longer interesting.

The experiment used in Chapter 4 to demonstrate how a design strategy can be developed which exploits the independence between the main design modes tests only a limited part of the model. We see that the independence described in the model appears in the behavior of the design teams. The building block design strategy is, however, obviously superior, even without having completed the work transformation matrix analysis. A more convincing experimental confirmation would demonstrate that the work transformation method could identify a superior strategy where the distinction is less obvious in the absence of such a tool.

(6) The fullest use of a model is to be able to apply its results to improve the behavior of the system under consideration. When a model is used to guide decisions within an organization it can be said to be used for control. The models in this thesis have not been used in this manner. The sequential iteration model is probably too limited to be able to be used for control. The work transformation model, on the other hand, does provide useful output, such as which sub-sections of a coupled design problem are somewhat independent, which can be potentially be used to partition the design problem into manageable pieces. Illustrating the effectiveness of this approach is beyond the scope of the current work.

For non-analyzable models it may be necessary to run simulations to obtain the results which are compared with data and observations. For simpler models, results are available through direct analysis of the mathematics in the model. Simulation leads to a greater degree of uncertainty about the validity of the model. The modeler cannot be certain
that the results and interpretation of the simulation are due to the particular data which were the inputs to the simulation, or if the model is more generally valid.

The models described in this thesis have been restricted to those where analytical results were available. This ensures the generality of the results if the assumptions are appropriate, but leads to a more restrictive set of models.

5.2. Iteration and the Models

There is a similarity between the two types of iteration described in section 1.3 and the two models of the iteration process described in Chapters 3 and 4. The probabilistic sequential iteration model in Chapter 3 is a description of unpredictable iteration (backtracking). The work transformation model in Chapter 4 is a description of the iteration due to complexity.

As we model the iteration process, we must recognize the distinct reasons for iteration, and ensure that the model we are using addresses the features of the design process in which we wish to apply our model. In particular, if we wish to capture the complexity of both types of iteration a hybrid model which encompasses both of the models in Chapters 3 and 4 might be necessary.

5.3. Implications for Engineering Management

If a model of the iteration process is to be useful to an engineering manager, it must provide information which enables the manager to make a better decision than would have otherwise been possible. There are potentially both direct and indirect benefits of building and using such a model. This section describes those benefits.
Despite the inaccuracies in a model, there are indirect benefits available through modeling. Having the designers or their managers aid in the construction of a model of their own behavior forces them to build a consensus as to what the important design tasks are, and how they will relate to the other tasks [Graham 1985]. Even if the model produces a description of the iteration process which does not correspond to the actual progress of the project, they will have derived benefit by considering the broader implications of their portion of the design on the entire process.

A further benefit of using the Design Structure Matrix as a modeling technique for engineering management is that it explicitly acknowledges and recognizes the iteration which is a dominant feature of engineering design. If managers attempted to use a PERT or other non-cyclic model of project management there would be poor correspondence between the model and actual behavior. Repetition of tasks, necessary in an iterative model, is viewed as non-productive in an acyclic model. Warfield [1990] argues that a systems analysis tool which recognizes where there are tasks which cannot be completed in sequence is useful to help participants expect cyclic behavior.

Also, the Work Transformation Matrix model is consistent with the criteria identified by Liberatore and Titus [1983] concerning the role that management science models can play in project management. Management science models are most relevant to managers when they present information to the manager which allows justification of managerial decisions. The Work Transformation Matrix model has its greatest effect in identification of which tasks are tightly coupled and will be the source of the greatest iteration. The manager is then able to assign coupled tasks to the engineering staff in such a way that the project will
have an increased chance of technical success. It is, however, unlikely that a manager will give over control of the project to the suggestions of any automated decision support system. Rather, such a system must be able to provide support for the decision making process.

5.4. Conclusion

This chapter has described ways that a model of the design iteration process can be verified and used. It is suggested that there is sufficient correspondence between the Work Transformation Matrix model of Chapter 4 and observations of design organizations that the model can serve as a useful tool for identifying the nature of design iteration. It may be possible to use this information to assign designers or to schedule design tasks in an efficient way, but the effect of using the model in such a way remains to be shown.

The probabilistic sequential iteration model of Chapter 3 has not been shown to have the same level of correspondence with observed organizations. It remains an interesting description of the engineering design process, but it is unlikely that this model will be as useful for managerial decision making.

There are potential benefits in using an iteration model, whether one of the two in this thesis or some other. An iteration model recognizes that there is cyclic behavior in the design process, and that the cyclic behavior affects other factors of interest, such as the amount of time that it takes to complete product development. Therefore, there is in principle utility of an iteration model in helping a manager or engineer to identify criteria which would otherwise remain invisible.
Chapter 6. Conclusion and Future Work

This thesis has identified two ways that engineering design iteration time can be modeled. These models incorporate numerical information into the Design Structure Matrix Method. These models are the Probabilistic Sequential Iteration Model and the Work Transformation Model.

Several different strategies to verify the models have been attempted. First, the Work Transformation Model of Chapter 4 has been applied to two industrial design problems. It is seen that the tasks which are highly iterative among the actual design process are the same as those which are indicated by the iteration model. In addition, the Delta Design Game was analyzed and shown to have two separable design modes; the strategy which exploited the independence between the design modes was shown to be a superior design strategy. The Work Transformation model has been shown to be able to identify the main features of a design process.

The Probabilistic Sequential Model of Chapter 3 is an interesting description of the iteration process, but has not been applied to industrial design settings. The information which is represented in the matrix is difficult to gather and the matrix is therefore difficult to implement in practice.

If the models presented in this thesis are to be useful to engineering managers, it must be shown that they are able to help the manager make better decisions. The models are currently able to diagnose and describe the iteration process, but have not been shown to improve the process by either shortening the process or improving the quality of the output design. The
recommendations that a model might make are those such as resource allocation decisions, or the scheduling of tasks to be performed.

One such decision that a manager might make is briefly described in the thesis. It involves whether product design and process design for the electronics module (section 4.5) should be completed simultaneously, or whether they should be done sequentially. It is shown that doing the design sequentially can save development costs (in terms of engineering hours), while doing the design in parallel can shorten the development cycle. It would be useful research direction to make this distinction more general, and to develop an automatic tool which would generate and evaluate possible parallel/sequential alternatives.

Also, it would be useful to develop the tools in this thesis so that they are easier to use. The data collection and analysis have been done by those who are familiar with both the DSM method and with the mathematics of the models. If the models are to be applied in a wider environment, the tools should be simpler. This would be the case if there were a computer program which suggested questions to be asked of the designers about the tasks and dependencies, which constructed and partitioned the matrices, and which did the mathematical analysis and was able to identify the important design modes. This kind of computer tool would not be difficult to develop, but does not currently exist.

**Future Work**

Applications of the models in this thesis are limited to those situations where the technological interactions are known in advance. It would be useful to be able to model the iteration process where the technological constraints are less well understood. This would require
significant change to the nature of the models, and remains a significant open question.

There are multiple possible manners of attempting to address the design process in a technologically novel environment. First would be to describe the design structure matrix at several different stages during the development process, in order to see how the structure of the solution strategy changes with time.

A second strategy would be to incorporate the tools of Marples [1961] and Luckman [1967] (described in the Chapter 1) with a Design Structure Matrix description of the design process. The DSM has the advantage of taking a holistic view of the problem, while the previous two methods are able to document individual design decisions. A joining of the techniques might be able to preserve the strengths of both methods.

Another direction in which this thesis research could be extended is to introduce the idea of resource constraints. Both of the models in this thesis assume that there are no such resource constraints to limit the design process (such as one designer who works on two projects, or two separate tasks which require the same computer resources). If there were such constraints then the models would become more complex. An engineering design organization could be viewed as a network of queues, where each design project is associated with a number of individual jobs within the network (an example of such an approach is Adler et al. [1992], described in section 1.4.2.) Because of the relationships between the jobs this type of network would prove difficult to analyze, although it is a more general description of an engineering design organization.

A further extension to this modeling paradigm would be to incorporate description of how the quality of the produced design is a
function of the iteration process. The models which are described in the thesis do not suggest that the final product design is a function of the way the iteration process is managed. A more general model would include effects of the iteration process on the product design. One recent thesis has developed a time and quality model of engineering design iteration [Pekar 1992]. This model is insightful in looking at time and quality simultaneously, but it limited to the particular product for which it was developed. It would be useful to develop a more general model which could be used for a wider variety of products.

All of these areas are ways in which this type of design process model can be extended to include the whole of the engineering design iteration process. This is a field which will provide ample opportunities for further research in coming years.
References


