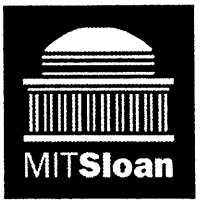


Massachusetts Institute of Technology

Sloan School of Management

Working Paper

Identifying Controlling Features of Engineering Design Iteration

Robert P. Smith
Steven D. Eppinger

Revised October 1995
Working Paper Number 3348

Forthcoming in *Management Science*

Acknowledgment

This research was funded by General Motors and by the Leaders for Manufacturing Program, a partnership involving thirteen major U.S. manufacturing firms and MIT's schools of engineering and management. The authors are also grateful to Steve Graves, Dan Whitney, Marcie Tyre, Karl Ulrich, Glen Urban, and two anonymous reviewers from *Management Science* who provided helpful and insightful comments on earlier versions of this paper.

Contact Addresses:

Prof. Robert P. Smith
Department of Industrial Engineering
University of Washington
Seattle, WA 98195

smith@ieng.washington.edu

Prof. Steven D. Eppinger
Sloan School of Management
Massachusetts Institute of Technology
Cambridge, MA 02142-1347

eppinger@mit.edu

Abstract

Engineering design often involves a very complex set of relationships among a large number of coupled problems. It is this complex coupling that leads to iteration among the various engineering tasks in a large project. The design structure matrix (DSM) is useful in identifying where iteration is necessary. The work transformation matrix model developed in this paper is a powerful extension of the DSM method which can predict slow and rapid convergence of iteration within a project, and predict those coupled features of the design problem which will require many iterations to reach a technical solution. This model is applied to an automotive brake-system development process in order to illustrate the model's utility in describing the main features of an actual design process.

Introduction

Manufacturing firms today face tremendous pressure to improve product development performance. Particular attention is paid to the time it takes to develop a new product or to redesign an existing one. Accordingly, there has been much written about product development performance and improvement [Clark and Fujimoto 1991; Wheelwright and Clark 1992; Smith and Reinertsen 1991; Whitney 1990; Blackburn 1991; Rosenthal 1992]. One approach to improve product development is to recognize that product development is often quite procedural and repeatable; therefore the process can be modeled in much the same way as we might for a manufacturing process that we wish to improve.

PERT charts, often created by managers to depict product development, describe the process as a progression of series and parallel activities; however, coupled and iterative product development processes are in fact quite common [Kline 1985]. Understanding design iteration is therefore fundamental to accelerating and improving product development practices. This paper presents a model of design iteration that can provide useful insights in this regard.

Examples of design iteration are not difficult to find in practice, yet are scarce in the published literature. At MIT, we have conducted a number of studies to document design iterations in industrial practice within the automotive and electronics industries [Chao 1993, Osborne 1993, Cesiel 1993, Marshall 1991]. Also, Clark and Fujimoto describe several cases where engineers trade essential technical information and thereby create rework for one another with each transfer [Clark and Fujimoto 1991]. Some researchers describe design iteration in terms

of interactions between design activities [Whitney 1990] or in terms of negotiation among technical specialists [Bucciarelli 1994].

There are two ways to accelerate an iterative development process: 1) to execute faster iterations, or 2) to conduct fewer iterations. Both approaches are important to consider. Faster iterations are achieved through several means, such as the use of engineering models or information technology. Fewer iterations may be experienced when the coupled development activities can anticipate each others' results, or when extraneous activities are removed from the iterative portion of the process.

Models of design iteration can provide valuable insight into the iteration process. The parallel iteration model presented here provides managers with information as to which activities in a complex and coupled process may be contributing the most to the iterative development process. Solutions to the iteration problem can also be developed using these models. Solutions may include adding resources, restructuring the process, providing new engineering automation tools, redefining the problems, limiting the scope of the development effort, reassigning tasks, etc.

Several researchers have recently begun to develop models of design iteration. Ha and Porteus [1994] present a model of coordination between two coupled development activities. They address the frequency of design reviews at which times information is transferred between the activities. Ahmadi and Wang [1994] describe a model of an iterative design process which is used to decide the composition and relation of design teams and to calculate the total amount of time needed during a design process. Hoedemaker *et al.*[1994] discuss iteration and rework as caused by project complexity due to the inefficiencies of interfaces between parallel tasks. Their models analytically show some of the theoretical limits to the concurrent engineering paradigm. Ford *et al.* [1993] describe a system dynamics model of the product development process. They show the effects on product development time and quality of various managerial inputs. Also, Krishnan *et al.*[1994] have created a model of overlapped development activities. Their model shows how to find the proper timing of the information transfers from upstream to downstream tasks in order to minimize total lead time. Further, we have developed a distinct model of sequential design iteration

[Smith and Eppinger 1994]. The sequential iteration model allows computation of the total lead time for a group of tasks where each task has a probability of creating rework for the other tasks. Each of these modeling efforts is an important contribution in this new and emerging area of management science because each model is able to explore a different facet of design iteration. The complexity of design iteration prohibits any one model from yet capturing all observed behaviors nor answering all questions of managerial interest.

Our work complements the above research by exploring the process of design iteration in greater detail. The model in this paper is concerned with "parallel iteration", an extreme case where a number of development activities are underway at one time. Each activity generates information which may cause the others to repeat all or some part of their own work. In this situation, we consider the repetition of activities to be deterministic and the entire set of activities converges to the design solution at once. We are able to analyze a large engineering design process and determine which subsets of tasks require the bulk of the expended effort during the iteration process.

We begin in the next section by introducing the design structure matrix as a method for modeling engineering processes. We then show how iteration can be modeled by extending this representation and we develop an analytical interpretation based on the eigenstructure of the matrix model. Next we apply this technique to the design of automotive brake systems. We conclude the paper with a discussion of this method's utility and limitations in understanding engineering design processes.

Modeling Engineering Design Processes

Engineering design is the process whereby a technical solution is developed to solve a given problem. There have been several attempts to give formal structure to the design process, such as those of Alexander [1964], Pahl and Beitz [1988], and Suh [1990]. This stream of research characterizes good design practice in general terms, but does not describe what makes some design problems more difficult than others. Our work extends the literature of design

process modeling by providing additional richness to the descriptions of design procedures so that the more iterative portions of design problems can be identified and strategies can then be developed to facilitate the effective execution of these difficult aspects.

The design structure matrix (DSM) serves as the basis for our formal analysis and will be briefly reviewed here. (For a more detailed overview of the DSM method and its applications, the reader is referred to Steward [1981] and to Eppinger *et al.* [1994].) The philosophy of the DSM method is that the design project is divided into individual tasks, and the relationships among these tasks can be analyzed to identify the underlying structure of the project.

It has been suggested that studying the relationships between individual design tasks can improve the overall design process, and is a powerful way to analyze alternative design strategies [von Hippel 1990]. Earlier work developed a modeling formalism which shows how different aspects of a design problem are related [Alexander 1964]. Alexander describes a graphical technique where the functional needs of the technology are nodes, and interactions between the needs are arcs. His idea is to segment the graph into subsections which have relatively few interactions across boundaries. These graph segmentations give rise to technical subsystems which should separate the technical needs into independently solvable problems. The DSM method is similar to Alexander's technique, but the nodes are now specific design tasks and the arcs are directed and indicate information flows between tasks. The nodes in the graph are arranged in a square matrix where each row and its corresponding column are identified with one of the tasks. Along each row, the marks indicate from which other tasks the given task requires input. Reading down each column indicates which other tasks receive its output. Diagonal elements do not convey any meaning at this point, since a task cannot depend upon its own completion. For example, Figure 1 shows a design structure matrix describing a simplified camera design procedure. In this process, task **C** requires input from tasks **B**, **D**, **E** and **F**, task **B** requires input only from task **A**, and task **A** needs no input to begin.

The DSM can be used to identify orderings of tasks and to identify difficult aspects of the design process. Some or all of the elements of the matrix can be made sub-diagonal (such as those

corresponding to tasks **A**, **B**, **G**, and **H** in Figure 1) by reordering the tasks of the matrix using a partitioning algorithm [Steward 1981, Rogers and Padula 1989, Gebala and Eppinger 1991]. An entirely sub-diagonal matrix indicates that there exists a sequence where all tasks can be completed with all input information available. Such a sequence may contain both tasks which must be done in series, and tasks which may be done in parallel. The information in a sub-diagonal design matrix is then similar to that expressed in a CPM (critical path method) or PERT (program evaluation and review technique) chart.

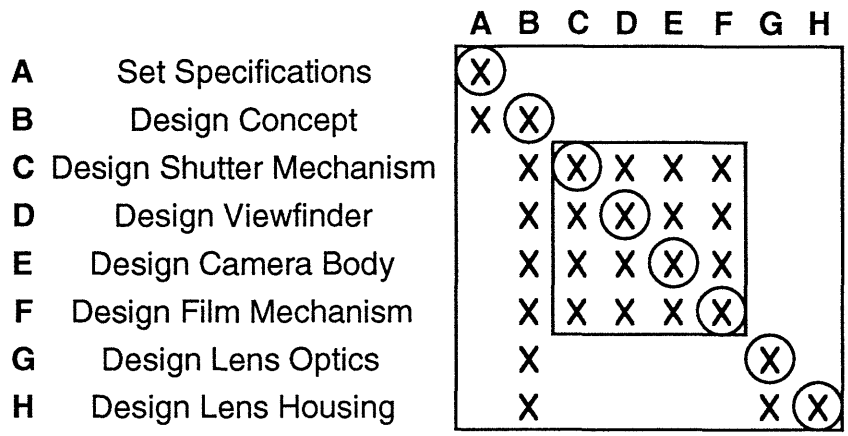


Figure 1. Sample Design Structure Matrix

More typically, due to the complexity in modern engineering design, the matrix cannot be reordered to have all matrix elements sub-diagonal (such as tasks **C**-**F** in Figure 1.) In these cases, there is a cyclic flow of information in the design process and standard CPM/PERT techniques are not applicable because of the presence of such cycles. Likewise, a sequential progression of the design tasks is not possible. Tasks where neither a purely sequential nor a parallel ordering is feasible are coupled in such a way that some alternative process for resolving the design interactions (such as iteration or negotiation) must be used. For this reason, iteration is a typical feature of engineering design projects [Hubka 1980]. The 4x4 sub-matrix in Figure 1 depicts a design problem defined such that the tasks are sufficiently complex and interrelated so that iteration will be necessary to complete the tasks.

It is useful to note that there is an established set of models which allow looping within a PERT modeling framework. This set of models is known as GERT (general evaluation and review technique). Taylor and Moore [1980] discuss the application of GERT to R&D projects. However, direct analysis of any but a simple GERT network is difficult, so simulation is typically used to evaluate a project [Neumann and Steinhardt 1979, p. 172]. It is the intention of our modeling effort to provide an analytically tractable model of the design iteration process, even for large projects. It is hoped that by preserving tractability it will be possible to observe the relationship between the structure of the problem and the development time of the project. Because GERT relies on simulation for large projects, it is difficult to discern this relationship.

There is evidence that competing companies faced with the same design problem may choose differing design strategies, which implies a different underlying design structure matrix. For example, to what extent a firm chooses to work on tasks sequentially versus overlapping may significantly affect development time [Clark and Fujimoto 1991, Krishnan et al. 1994]. Furthermore, the way a firm decomposes the development problem into smaller problems has a major impact on the process [von Hippel 1990], and therefore changes the DSM. Since a DSM model is specific to one firm's process, it can be a powerful tool in reengineering today's product development procedures.

For our purposes, we assume that the tasks and interrelationships of a design problem are known and unchanging during the course of the project. This assumption is reasonable for a firm is working on a design project in an area in which they have a significant degree of familiarity. (The example of brake-system design at General Motors, which serves as the basis for the application described in this paper, fits this category.) The assumption is less true for a completely new or rapidly evolving technology.

Development time is an important measure in engineering design management. We believe that complex iteration is a major source of extended development time. While the design structure method is a useful tool to identify the coupled blocks in which the complex iteration occurs, our work is intended to characterize how such iteration occurs.

If we include task durations in the DSM, we can use this description to estimate the total duration of the project. Series tasks can be evaluated by summing their individual times, and parallel tasks can be evaluated by finding the maximum of those task times. For the project characterized by the DSM in Figure 1, if the task times are a, b, c, ... , h, the time of the camera design project would be

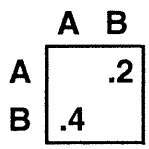
$$a + b + \max\{ f(c,d,e,f) , g+h \}$$

where f(·) is an unknown function corresponding to the development time for the coupled block.

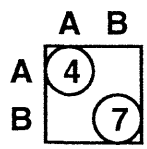
The model presented in this paper illustrates how iteration time can be evaluated for such a coupled block of tasks, and how to identify which aspects of the design problem contribute the most to iteration time. The model relies on standard linear algebra results, yet the interpretation of the relationship between the matrix structure and development time is novel.

Design Iteration Model Development

To model design iteration, we use an extended numerical version of a fully coupled design structure matrix which we call the work transformation matrix (WTM). There are two types of information in a work transformation matrix. The off-diagonal elements (Figure 2a) represent the strength of dependence between tasks, giving rise to the transfer of work, or rework, involved in the iterations, as defined in next section. The diagonal elements (Figure 2b) in the WTM represent the time that it takes to complete each task during the first iteration. It is assumed that there will be multiple iterations, and that the time a task requires for each subsequent stage is a function of the amount of time spent working in the previous stage on tasks which provide its input. We wish to find the sum of the times of all stages.



(a) Strength of Dependence Measures



(b) Task Times

Figure 2. Work Transformation Matrix

For the purposes of our analysis, we assume that each task creates a deterministic amount of rework for other tasks. Rework is the required repetition of a task because it was originally attempted with imperfect information (assumptions). Rework therefore adapts the evolving solution to account for the modified information.

The derivation below is divided into three sections. In the first section we describe the assumptions and mechanics underlying the model. In the second section we describe why the eigenvalues and eigenvectors of the WTM are relevant to our analysis of development time. In the third section we describe how the eigenstructure of the matrix is interpreted. Following the derivation of the model, we illustrate the analytical process using a simple example, and then we analyze the WTM for the brake-system design problem.

A: Work Transformation Matrix Model Assumptions

The WTM model makes three particular assumptions which allow us to perform linear algebraic analysis of the WTM:

- All tasks are done in every stage – fully parallel iteration.
- Rework performed is a function of the work done in the previous iteration stage.
- The work transformation parameters in the matrix do not vary with time.

The first assumption, that all coupled tasks are worked on in every stage, is an idealization of an observation we have made concerning many design projects. The assumption would be likely to hold in a situation where there is a team whose membership is fixed, who are geographically close, and who are working on a set of interrelated design issues simultaneously. This situation is commonly recommended for concurrent engineering practice. In such an environment, we are concerned with the tightly coupled design tasks (not the sequential or decoupled ones) for which the fully parallel iteration assumption seems reasonable. It is important to note that this tightly coupled portion of the development process may not include the entire product development process, but an iterative subset of the process, as is the case for the brake-system design example presented in this paper. If there are organizational or temporal barriers

which do not allow for simultaneous work in close cooperation, then this assumption would likely not hold. For a relaxation of the fully parallel assumption, we have also considered an iteration model in which tasks are executed in multiple phases [Smith and Eppinger 1995].

The second and third assumptions, that rework is a linear function of work in the previous iteration and that the rework proportions do not vary with time, are convenient from a mathematical standpoint but more difficult to argue from an empirical perspective. Nevertheless, it has been observed that the amount of time per iteration does decrease (designs converge over time) [Tjandra 1995]. Furthermore, if the number of iterations is relatively small (the design process is very stable), most of the rework is completed in the first few iterations. So if the parameters were to change over time, then the changes would primarily affect the amount of work in iterations that do not contribute strongly to the total time required. Given these observations, the second and third assumptions seem reasonable as an abstraction.

To describe the model, we first introduce the concept of the work vector u_t. This is an n-vector, where n is the number of coupled 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, refer to Appendix A.) However, work on each task will cause some rework to be created for all of the other tasks that are dependent on the completed task for information. The work transformation matrix documents such information dependence. 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 then the strength of dependencies portion of the

WTM (Figure 2a). The diagonal entries of A are zeros. Since the process is iterative, the work vector u_t can be also be expressed as:

$$u_t = A^t u_0$$

The sum of all of the work vectors is the total work vector U, representing the total number of times that each of the tasks is attempted during the total of M iteration stages of the design process:

$$U = \sum_{t=0}^{M} u_t = \sum_{t=0}^{M} A^t u_0 = \left(\sum_{t=0}^{M} A^t \right) u_0$$

The model output U is therefore in units of the original amount of work done on each task in the first iteration stage. (If element i in vector U is 1.6, then the design organization will have done a total of 60% rework on task i in subsequent stages.) For a time-based interpretation of the matrix A, see Appendix B. For now, 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 (Figure 2b), then WU is a vector which contains the amount of time that each task will require during the M iteration stages.

B: Eigenvalue Decomposition

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

$$A = S\Lambda S^{-1}$$

where Λ 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:

$$A^t = S\Lambda^t S^{-1}$$

The total work vector U can therefore be expressed as:

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

If the magnitude of the maximum eigenvalue is less than one, then the design process will converge (i.e. as M 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 in some task during an iteration stage will create more than one unit of work for that task at a future stage. Such a system is unstable and the vector U will not converge, instead growing without bound as M increases. (It is a sufficient, but not necessary, condition for stability that the entries either in every row or in every column of matrix A sum to less than one.)

If we take the limit as M approaches infinity we can use the formula:

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

to obtain

$$U = S(I - \Lambda)^{-1} S^{-1} u_0$$

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 reach a solution. This situation is not likely to occur in concurrent engineering design environments where collaboration is encouraged, or in routine design processes where a team is responsible for bringing out a new variation of an existing product. If a design process is not converging, it might be appropriate to abandon the project, or to adapt the specifications and restructure the problem so that the process becomes stable. These types of changes are outside the approaches to the design process which are considered by the model.

The remainder of our discussion of the work transformation matrix model is limited to problems where a technical solution can be found in finite time (i.e., all eigenvalues are less than one). 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.

C: 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.

We use the term *design mode* to refer to a group of design tasks which are very closely related such that working on any one of them creates significant work, directly or indirectly, for each of the other tasks within the mode. Some of the design modes may correspond to important and/or recognizable subproblems within the coupled set of tasks. We use the eigenvalues and eigenvectors of matrix A to identify the design modes.

The magnitude of each eigenvalue of A identifies the geometric rate of convergence of one of the n design modes [Strang 1980]. 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 the given rate.

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 determining the decay or oscillation of the mode). The eigenvectors identify the mode shapes of natural motion, quantifying the participation of each of the state variables in each mode [Ogata 1967]. The use of eigenstructure analysis in order to observe primary behavior of a complex system is also used in many other fields. Some examples are from the fields of botany [Weber and Campbell 1989], geology [Tromp 1993], medicine [Throne and Olson 1994], chemistry [Procyk and others 1992], finance [Brown 1989] and biology [Tirion and ben Avraham 1993].

Based on the definition of the work transformation matrix we know that the matrix will be coupled and non-negative. 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.

The slowest design mode (largest eigenvalue) will therefore have an eigenvector which is strictly positive. This design mode gives us little problem with interpretation. The larger the element in its eigenvector, the more strongly that element contributes to that mode. Other design modes are, however, less obvious. Also by the Perron-Frobenius Theorem, there is only one eigenvector which is strictly positive. We must therefore be able to interpret negative and complex numbers in the eigenvectors as well as negative and complex eigenvalues.

Recalling that the total work vector U is calculated by

$$U = S(I - \Lambda)^{-1}S^{-1}u_0$$

we will look at the above formula for U to see how the eigenvalues and eigenvectors of matrix A can be used to interpret the design modes.

The Eigenvalues

The term $(I - \Lambda)^{-1}$ is a diagonal weighting matrix where each entry along the diagonal corresponds to one of the eigenvalues and has the form $1/(1-\lambda)$ where λ is an eigenvalue. To interpret the real eigenvalues, note that the function $1/(1-\lambda)$ is strictly increasing over (-1,1). A graph of this function is shown in Figure 3. We interpret this function to mean that the modes with larger positive eigenvalues have a greater contribution to the total work than do modes with smaller and negative eigenvalues. Therefore, as we consider which are the more important design modes, we restrict our attention among real eigenvalues to the larger positive eigenvalues.

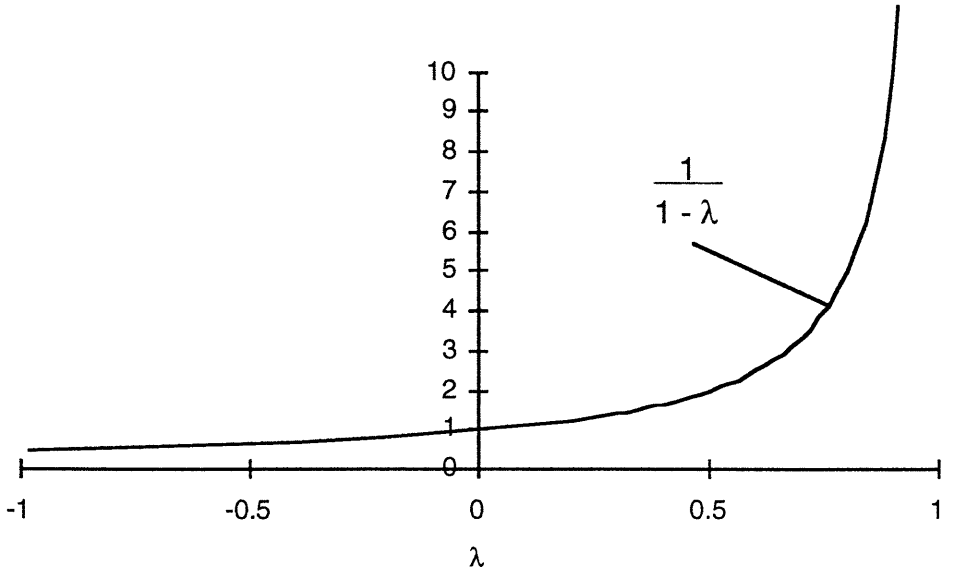


Figure 3. Graph of Magnitude vs. λ for Real Eigenvalues

For complex eigenvalues, we also wish to find the magnitude of the term $1/(1-\lambda)$. For a complex eigenvalue $\lambda = \alpha + \beta i$, we have:

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

We find an upper bound using the fact that $\beta \neq 0$:

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

Also, we can find a lower bound using the fact that $\alpha^2 + \beta^2 < 1$:

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

The graph of these upper and lower bounds is shown in Figure 4.

We see that the real part of complex eigenvalues gives bounds on the magnitude of the $1/(1-\lambda)$ term 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 these arguments we need only consider those eigenvalues which have a relatively large positive real component, whether they are real or complex.

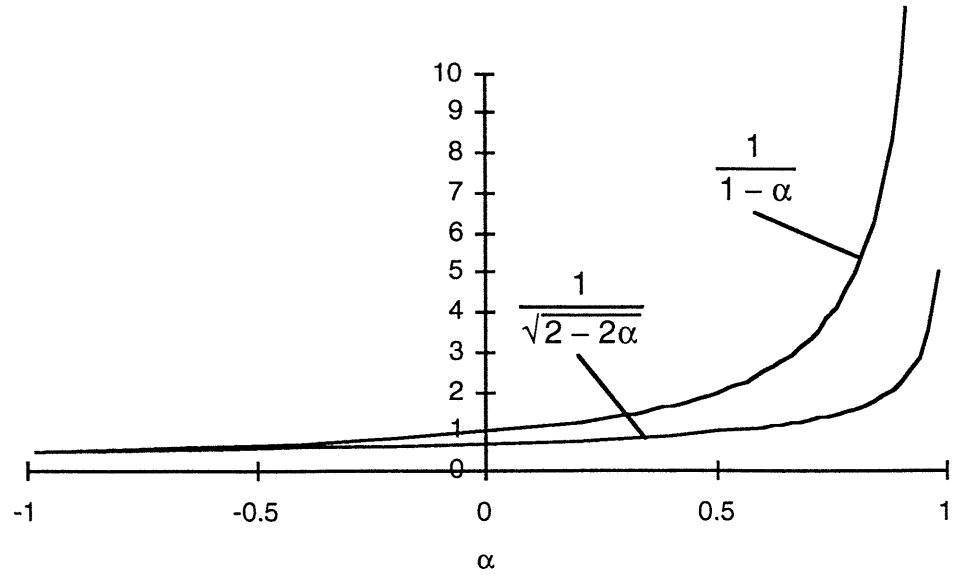


Figure 4. Graph of Bounds on Magnitude vs. α for Complex Eigenvalues

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 the 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.

The Eigenvectors

This section discusses the interpretation of the relative importance of each task within an eigenvector, given that we know the eigenvalue corresponding to that design mode. The final two terms in the formula for the total work are $S^{-1}u_0$. These terms combine to a vector, the elements of which provide (along with the $(I - \Lambda)^{-1}$ term described earlier) another weighting for each eigenvector. This weighting is both a magnitude and a direction.

Since the rows of S^{-1} corresponding to real eigenvalues are real, each weight for a real eigenvector is also real. Therefore, its 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. One way to consider the interaction is to calculate the contribution of the mode to the total work vector U and see which tasks give large contribution to the total work. (This method is similar to the method of selective modal analysis [Perez-Arriaga *et al.* 1990].)

We can compute the participation of the ith design mode by looking at the vector

$$S_{\bullet i} \frac{1}{1-\lambda_i} \sum_{j=1}^{n} S_{ij}^{-1}$$

where $S_{\bullet i}$ is the ith column of S. (Recall that u_0 is a vector of ones.) This vector shows the contribution of work from the ith design mode to the total work vector U. (Note that this vector is a scaling of the ith eigenvector when the ith eigenvalue is real; it is both a scaling and a change of direction for complex eigenvalues.) In practice this is a more difficult way to analyze the modes because of the possible ill-conditioning of S, however this vector is useful in analyzing complex eigenvectors when the associated eigenvalue has a large positive real part.

Ranking the Modes

It is possible to rank the modes using any of three measures. The first is to use the magnitude of the terms of $(I-\Lambda)^{-1}$. The second is to use the terms of $(I-\Lambda)^{-1}S^{-1}u_0$. The third is to look at how much each mode actually participates in the total work vector U, as described just above. In principle, all three of these methods can determine the relative importance of each design mode and of each design task within the design modes. In practice, even when S is invertible it may be ill-conditioned. (The matrix S in the brake-system example described later is ill-conditioned.) The ill-conditioning may lead to erroneous interpretations of modes which are not important contributors to the total work vector U. For this reason we suggest ranking of the design modes using the $1/(1-\mathrm{Re}(\lambda))$ terms (the first method).

A Simple Example

As an illustration of the above interpretation of eigenvalues and eigenvectors, 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 1. The tasks in this matrix are, in order: Design Shutter Mechanism, Design Viewfinder, Design Camera Body, and Design Film Mechanism. The numbers can be interpreted as follows: if the shutter is completely redesigned, then 30% of the viewfinder design work must be redone (entry in row 2, column 1 is 0.3), 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 (Λ) 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, and the associated eigenvalues are the diagonal elements of Λ. 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.

$$u_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad u_1 = \begin{bmatrix} 0.6 \\ 0.9 \\ 0.9 \\ 0.4 \end{bmatrix} \quad u_2 = \begin{bmatrix} 0.39 \\ 0.62 \\ 0.53 \\ 0.33 \end{bmatrix} \quad u_3 = \begin{bmatrix} 0.267 \\ 0.395 \\ 0.390 \\ 0.207 \end{bmatrix} \quad u_4 = \begin{bmatrix} 0.180 \\ 0.278 \\ 0.249 \\ 0.144 \end{bmatrix}$$

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 eigenvector of the most slowly converging eigenvalue dominates

the shape of convergence of the work vectors. It is also true that the associated eigenvalue governs the geometric rate of convergence.

We can understand even more by computing the total amount of work completed during the iteration process and by looking at the intermediate calculations. Inspection of the term $(I - \Lambda)^{-1}$ shows that the one positive eigenvector contributes significantly more work to the process than do the negative and the complex modes:

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

The total weight on the eigenvector matrix determines how each mode contributes to the total work:

$$(I - \Lambda)^{-1}S^{-1}u_0 = \begin{bmatrix} 6.513 \\ -0.223 \\ 0.093 - 0.399i \\ 0.093 + 0.399i \end{bmatrix}$$

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 this primary design mode.

We are now able to calculate the total work vector:

$$U = S(I - \Lambda)^{-1}S^{-1}u_0 = \begin{bmatrix} 2.807 \\ 3.755 \\ 3.595 \\ 2.375 \end{bmatrix}$$

There has been more work completed during the process by the middle two tasks, as indicated by the preliminary inspection of the eigenvectors and eigenvalues.

Application to Brake-System Design

In order to verify the utility of the work transformation matrix technique, we now demonstrate the analysis of an actual design process and show the types of insights available. 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 to the iterative portion of the brake-system design process. In preparing this analysis, we followed up Black's

work by spending several months doing field work at the brake-system design facility of General Motors. Our field work included informal discussions with systems and component engineers, examination of internal documentation, and interviews with engineers and their managers.

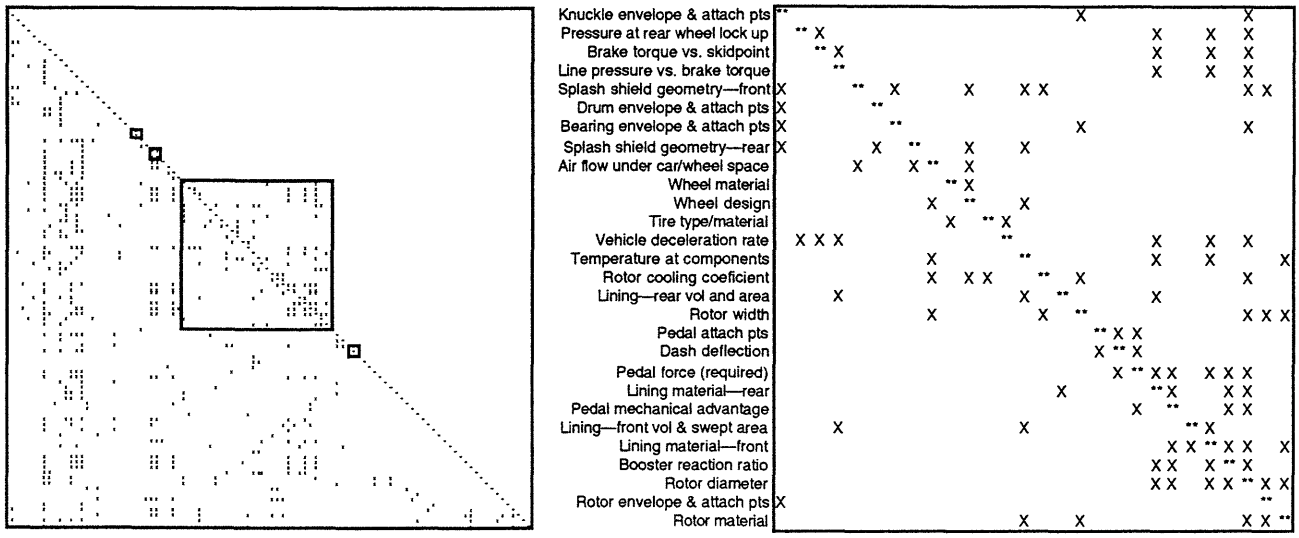
We have found the brake system to provide a good subject for modeling of the design process because of the nature of the design problem. Brake-system design is stable in that the technology and the market are mature and the form of the base product is not undergoing radical change. The brake-system design engineers have considerable experience with brake-system design. These factors suggest that the data contained within the brake-system DSM are not changing rapidly, and the knowledge represented within the DSM is well developed.

There are four questions which must be answered in constructing the work transformation 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 model includes data of the first three types, but does not include any explicit time data. Many of the observations about the controlling features of the design process can be made without having the time data available. In particular, we are able to identify the total number of iterations taken on each task.

In modeling the brake-system design process, Black found several design activities to be tightly coupled. Further investigation of each task identifies the actual design parameters involved in each task and how they are interrelated. This information is captured in the brake-system DSM from Black *et al.* [1990], shown in Figure 5a. Analysis of the matrix reveals that the problem can be partitioned into a block of complex, coupled design parameters at the center of the matrix, preceded by and followed by groups of sequential and parallel parameters. The coupled, iterative block is expanded in Figure 5b. (We realize that Figure 5a is too small to see the details of the matrix; it is included here to illustrate the overall structure of the DSM, which involves over 100 design parameters of which 28 form the iterative sub-problem we will study further.)

Although parameter interactions are captured by the matrix, these interactions are generally not well understood by those executing the development process, a situation which leads to excessive development time. If all interactions were well understood on a technical level, then the brake-system performance 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.



(a) Complete Matrix **(b) Expansion of Iterative Block**

Figure 5. Brake-System Design Matrix

The customer wants an automobile with quiet, smooth brakes that do not require frequent service. To the design engineers this means that the 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 respectively 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 (noise) 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. These problems are believed to be inherent consequences of using dry friction to stop a vehicle. Prior to our analysis, the design engineers held these three problems (noise, pulsation and wear) to be the "controlling features" of the design/test/redesign iteration problems which they were experiencing.

Using the Work Transformation Matrix Method to Identify Controlling Features

To perform our analysis, we translate the coupled portion of the binary DSM (Figure 5b) 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 6.) Each off-diagonal value is an estimate of the amount of work (as a percent of the amount of time that it took to determine the parameter during the original iteration) that the upstream task creates for the downstream task. The engineers in the design organization were asked to describe why each piece of information was necessary and to give the relative importance of each of the pieces of input information. We then assigned numerical values to the dependencies described by the engineers. 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 modes is robust against minor changes in the values entered in the matrix. This robustness can be demonstrated in two ways: (1) If we scale all of the values in A by a constant factor, the eigenvectors will be unchanged. The eigenvalues will scale proportionally, and our interpretation of the analysis will not change. (2) If we scale only one set of values (say strong dependence becomes 0.6 instead of 0.5), then there would be no significant changes to the resulting eigenstructure. More details on sensitivity of the eigenvectors to the weights are given in [Smith 1992].

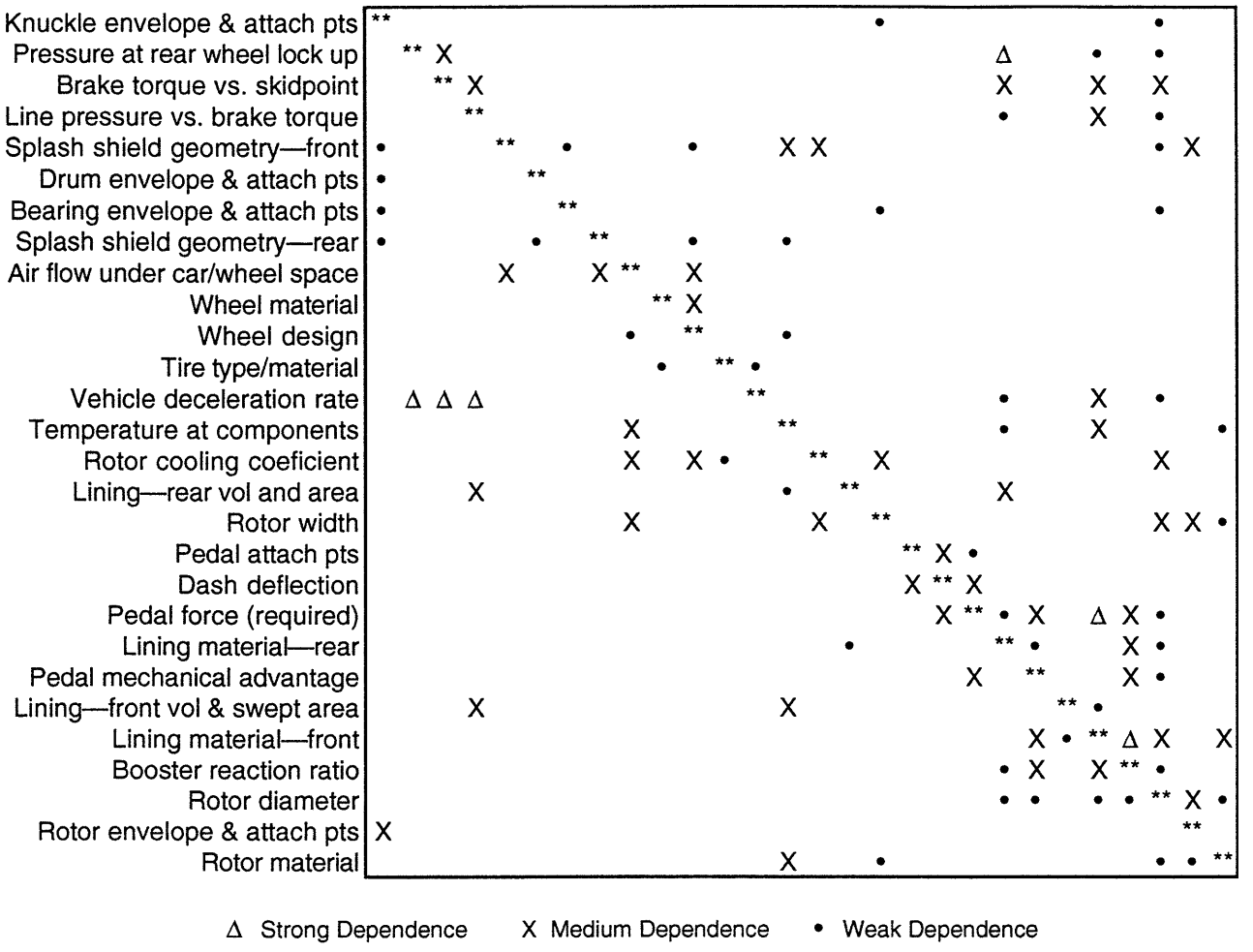


Figure 6. Coupled Block from Brake Matrix with Weighted Dependencies

The design modes which dominate the iterative rework process are represented by the eigenvectors of the work transformation matrix corresponding to the largest eigenvalues. Figure 7 shows the value of the quantity

$$\frac{1}{1 - \mathrm{Re}(\lambda)}$$

for each eigenvalue λ. This value is an upper bound on the magnitude of the effective number of times each design mode will be repeated during the iteration process (as described earlier).

Table 1 shows the values of the elements in the first two eigenvectors, as well as the total work vector U. (The larger magnitude elements are highlighted for emphasis and the eigenvectors

are scaled to be unit vectors. Only the first two design modes are shown because they are sufficient to characterize the larger magnitude elements in the total work vector.)

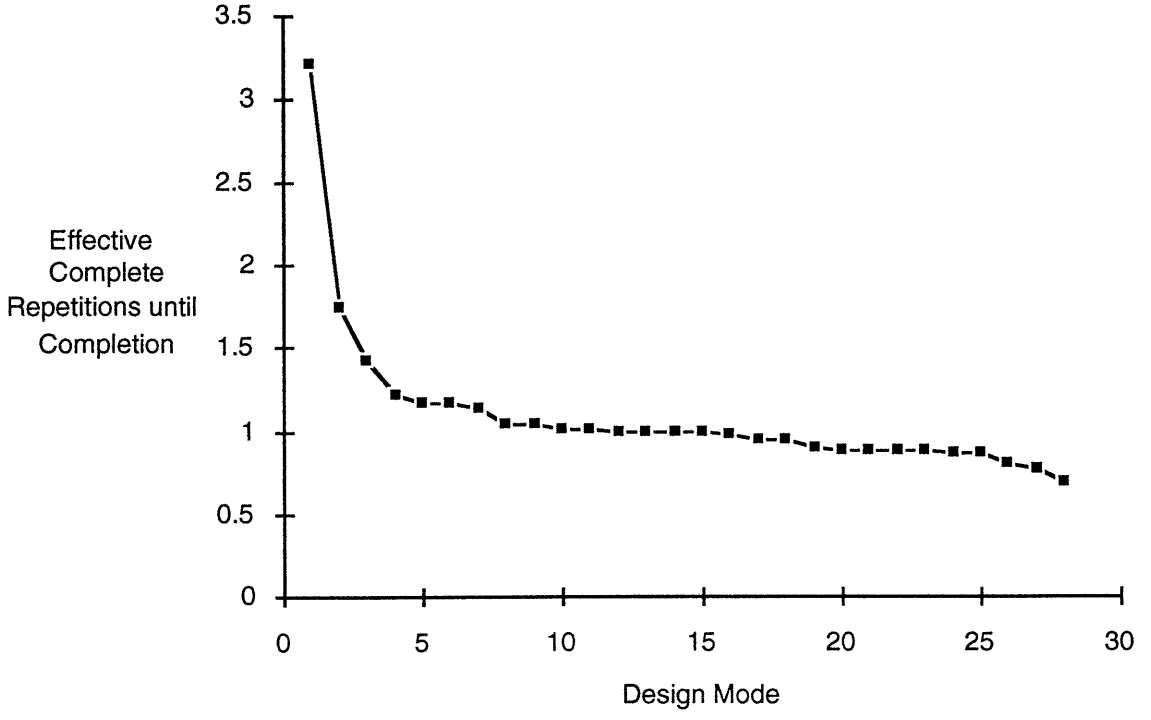


Figure 7. Brake-System Eigenvalues

The first design mode, represented by the first column in Table 1, is primarily composed of Vehicle Deceleration Rate and Pedal Force Required, with lesser involvement from Pressure at Rear Wheel Lockup, Brake Torque vs. Skidpoint, Dash Deflection, Pedal Mechanical Advantage, Front Lining Material, and Booster Reaction Ratio. This design mode identifies the group of design parameters requiring the greatest amount of work before convergence on the final acceptable design. We interpret this mode as the "stopping distance problem". Solving this problem assures

Parameter Name	First Design Mode	Second Design Mode	Total Work Vector
Knuckle envelope & attach pts	0.0106	0.0616	1.2764
Pressure at rear wheel lock up	**0.1916**	-0.0290	**3.3924**
Brake torque vs. skidpoint	**0.2271**	0.0090	**3.8290**
Line pressure vs. brake torque	0.1275	0.0094	2.3748
Splash shield geometry—front	0.0875	**0.4519**	**3.2229**
Drum envelope & attach pts	0.0008	0.0070	1.0638
Bearing envelope & attach pts	0.0113	0.0686	1.3402
Splash shield geometry—rear	0.0120	0.0374	1.3300
Air flow under car/wheel space	0.0407	**0.3102**	2.4563
Wheel material	0.0048	0.0330	1.3181
Wheel design	0.0132	0.0582	1.2722
Tire type/material	0.0381	0.0034	1.4245
Vehicle deceleration rate	**0.5225**	-0.0030	**7.1728**
Temperature at components	0.1415	**0.2032**	**2.9880**
Rotor cooling coefficient	0.0751	**0.5172**	**3.3854**
Lining—rear vol and area	0.0970	0.0088	2.2920
Rotor width	0.0739	**0.5216**	**3.4214**
Pedal attach pts	0.1167	-0.0850	2.0459
Dash deflection	**0.2229**	-0.1240	**2.9961**
Pedal force (required)	**0.4998**	-0.1330	**5.9383**
Lining material—rear	0.1126	-0.0350	2.1957
Pedal mechanical advantage	**0.2669**	-0.1000	**3.3959**
Lining—front vol & swept area	0.1201	0.1226	2.5726
Lining material—front	**0.3158**	0.0190	**4.6385**
Booster reaction ratio	**0.2241**	-0.0470	**3.2237**
Rotor diameter	0.0723	0.0219	2.1069
Rotor envelope & attach pts	0.0038	0.0349	1.3191
Rotor material	0.0620	**0.1807**	2.0894

Table 1. Brake-System Eigenvectors and Total Work Vector

that the brake system will stop the car without creating uncontrollable skidding. A proprietary performance simulation for this problem has been developed and is a good predictor of actual

performance. These iterations can therefore occur quickly. Use of the simulation tool accelerates these iterations so a large number of iterations on the first design mode no longer strongly affects the total time of the development process. The model nevertheless confirms that the stopping performance problem is the fundamental controlling feature which affects design iteration.

The second design mode is composed of primarily Splash Shield Geometry, Airflow under Car/Wheel Space, Rotor Cooling Coefficient, and Rotor Width, with lesser involvement from Temperature at Components and Rotor Material. All of these factors are technical parameters corresponding to overheating and cooling of the brake system. We interpret this second design mode as a "thermal problem" related to the problems of noise, pulsation, and wear. For these problems, as described earlier, there are few analytical or simulation tools available to the design team. Many iterations are therefore required to converge upon a design solution, and there is no guarantee that those iterations can be rapid. Field or laboratory testing must be conducted to eventually reach a solution which meets the performance criteria, however these iterations are quite slow.

The final column in Table 1 shows the total work vector U for the brake-system matrix. We see that the largest terms in U are also the largest terms in the first two eigenvectors. This confirms that the ranking and interpretation of the eigenvectors gives useful information for identifying the structure of the total work vector for the brake-system design problem.

Using the work transformation matrix to compute the design modes enabled us to identify groups of design features which require significant numbers of iterations. Interpreting each mode requires knowledge about the underlying physical phenomena. Our analysis of brake-system design was able to identify the two dominant controlling features (stopping distance and thermal problems).

As a result of our work with General Motors, their brake-system design group has launched a substantial research effort to build better analytical models of the thermal performance of brake systems. A robust engineering model of the thermal issues may be able to speed the iterations now understood to be inherent in the process. Knowledge about these controlling

features can potentially be used to modify or control the design process in other ways as well, as described in the following section.

Discussion and Conclusion

The model presented here describes the underlying behavior of complex design iteration processes. These controlling features determine not only the duration of the product development process, but also the variations in development lead times. As managers and engineers well know, process improvement is facilitated by process understanding. Models of iteration provide essential understanding of the effects of complex relationships within development processes.

Our example of brake-system design shows that the work transformation matrix model is able to identify the "controlling features", or design modes. These features account for the bulk of the time taken in the iteration process. This identification is an important step in improving and shortening the development process.

Each design mode identifies an important sub-problem in the iterative development process. Design teams can then focus on improving their abilities to solve each sub-problem efficiently. The brake-system engineers have already "solved" the stopping distance design problem such that it does not require a significant amount of time to come to a solution. Our analysis creates additional incentive for the engineers to work on the thermal problems in the hope of improving the iterative product development process.

Once the most iterative design modes have been identified, there are several ways that development teams can go about accelerating the process. We recommend that teams consider two general strategies: faster iterations and/or fewer iterations.

Faster iterations can be achieved in a number of ways. These could involve the introduction of process improvements like the following:

- computer-aided design systems which accelerate some of the individual design tasks
- engineering analysis tools such as simulation techniques which reduce the need for time-consuming prototype/test cycles

- information systems involving database management and networking software which facilitate rapid exchange of technical information among individuals on the design team
- removing extraneous activities from the iterative process

Fewer iterations could be achieved by, for example:

- improved coordination of individuals whose work depends on one another
- co-location of team members responsible for tightly coupled activities, allowing faster and more frequent information exchanges and faster resolution of conflicting issues
- minimization of team size, which allows a core set of individuals to work more efficiently
- proper specification of interfaces, allowing for reduced need for interactions between individuals and teams within the development process
- use of engineering models capable of predicting performance along multiple dimensions, eliminating the need for separate analyses

The improvements which create faster iterations primarily involve shortening the times for each task, or changing the W matrix. The improvements for achieving fewer iterations primarily involve changing the rework quantities, or changing the A matrix. To implement any of the above methods, it is useful to know which portions of the process are most tightly coupled. Such understanding is facilitated by analysis of the work transformation matrix model presented here.

The metrics produced by the model represent the total work done by each of the tasks during the iteration process. These are useful metrics for comparing alternative policies (alternative work transformation matrices). However, the metrics by themselves are not able to suggest improved policies. Instead, it is useful to be able to examine how the iterative work is generated, which is done by examining the design modes, as presented in this paper. Identifying and implementing improvements relies on the insights given by the design modes as well as technical knowledge relevant to the problem domain.

We suggest that there is an important class of problems which are sufficiently well understood such that the engineers and managers involved can identify the tasks and the

information dependencies (the information necessary to construct the matrix), without being able to identify the controlling features of the overall problem. This is the class of problems for which a structured analysis is relevant and useful. The brake-system design problem fits into this class. It is also important to note that the model is situation specific. Our analysis of the brake-system design process at General Motors may not apply to another firm developing automotive brakes.

The question remains open whether it is possible to reliably generate the necessary matrix data for a problem with which the design organization has less familiarity. Nevertheless, applying the model to a more novel problem would likely provide new knowledge to the organization, allowing them to identify the critical issues prior to beginning the design process.

The work transformation matrix method as presented in this paper is a deterministic model. In realistic situations not all facets of a product development process are predictable. Opportunities exist to generalize this model to allow for stochastic elements. These generalizations may reduce the tractability of the model; a balance must be struck between tractability and realism. Randomness could be introduced to the WTM model along several dimensions that researchers have considered in other types of design process modeling, such as task duration [Nukala *et al.* 1995], rework ratios [Smith and Eppinger 1994], queueing delays [Adler *et al.* 1995], the possibility of project failure [Smith 1995], or other factors. These extensions remain as future work.

The work transformation matrix can serve as a useful modeling 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 suggest that improved simulation of the thermal aspects of the design problem may accelerate solution development. In general, the identification of the controlling features provides a crucial piece of information which enables a design team to better allocate resources in order to lessen development time.

References

Adler, Paul S., Avi Mandelbaum, Viên Nguyen, and Elizabeth Schwerer, "From Project to Process Management: An Empirically-based Framework for Analyzing Product Development Time," *Management Science*, Vol. 41, No. 3, pp. 458-484, 1995.

Ahmadi, Reza H., and Hongbo Wang, "Rationalizing Product Design Development Processes," Working Paper, Anderson Graduate School of Management, UCLA, 1994.

Alexander, Christopher, *Notes on the Synthesis of Form*, Harvard University Press, Cambridge, 1964.

Black, Thomas A., "A Systems Design Methodology Applied to Automotive Brake System Design," Masters Thesis, MIT Departments of Management and Mechanical Engineering, May 1990.

Black, Thomas A., Charles H. Fine, and Emanuel Sachs, "A Method for Systems Design Using Precedence Relationships: An Application to Automotive Brake Systems," MIT Sloan School of Management Working Paper #3208, October 1990.

Blackburn, Joseph D., *Time-based Competition: The Next Battleground in American Manufacturing*, Business One Irwin, Homewood, IL, 1991.

Brown, Stephen J., "The Number of Factors in Security Returns," *Journal of Finance*, Vol. 44, No. 5, pp. 1247-1262, 1989.

Bucciarelli, Louis L., *Designing Engineers*, MIT Press, Cambridge, 1994.

Cesiel, Douglas S., "A Structured Approach to Calibration Development for Automotive Diagnostic Systems". Masters Thesis, MIT, June 1993.

Chao, Linda, "Improving Quality and Time to Market in the VLSI Product Life Cycle", Masters Thesis, MIT, June 1993.

Clark, Kim B., and Takahiro Fujimoto, *Product Development Performance: Strategy, Organization, and Management in the World Auto Industry*, Harvard Business School Press, Boston, 1991.

Eppinger, Steven D., Daniel E. Whitney, Robert P. Smith, and David A. Gebala, "A Model-based Method for Organizing Tasks in Product Development," *Research in Engineering Design*, Vol. 6, No. 1, pp. 1-13, 1994.

Ford, David, Alex Hou, and Don Seville, "An Exploration of Systems Product Development and Gadget Inc.," Working Paper, MIT Sloan School of Management, 1993.

Gebala, David A., and Steven D. Eppinger, "Methods for Analyzing Design Procedures," ASME Design Theory and Methodology Conference, Miami, pp. 227-233, September 1991.

Ha, Albert Y., and Evan L. Porteus, "Optimal Timing of Reviews in Concurrent Design for Manufacturability," Yale School of Organization and Management Working Paper No. 1184 Rev., 1993, forthcoming in *Management Science*.

Hoedemaker, Geert M., Joseph D. Blackburn, and Luk N. Van Wassenhove, "Limits to Concurrency," Vanderbilt University Owen School of Management Working Paper #95-33, 1994.

Hubka, Vladimir, *Principles of Engineering Design*, Butterworth Scientific, London, 1980.

Kline, Stephen J., "Innovation Is Not a Linear Process," *Research Management*, Vol. 28, No. 4, pp. 36-45, 1985.

Krishnan, Viswanathan, Steven D. Eppinger, and Daniel E. Whitney, "A Model-Based Framework to Overlap Product Development Activities," MIT Sloan School of Management Working Paper #3635, Revised December 1994, forthcoming in *Management Science*.

Marcus, Marvin, and Henryk Minc, *A Survey of Matrix Theory and Matrix Inequalities*, Allyn and Bacon, Boston, 1964.

Marshall, David A., "Dynamic Benchmarking: A Comparative Study of Automotive Suppliers", Masters Thesis, MIT, June 1991.

Neumann, Klaus, and Ulrich Steinhardt, *GERT Networks and the Time-Oriented Evaluation of Projects*, Springer Verlag, New York, 1979.

Nukala, Murthy V., Steven D. Eppinger, and Daniel E. Whitney, "Generalized Models of Design Iteration using Signal Flow Graphs," ASME Design Theory and Methodology Conference, Boston, pp. 413-422, September 1995.

Ogata, Katsuhiko, *State Space Analysis of Control Systems*, Prentice Hall, Englewood Cliffs, NJ, 1967.

Osborne, Sean M., "Product Development Cycle Time Characterization Through Modeling of Process Iteration", Masters Thesis, MIT, June 1993.

Pahl, Gerhard, and Wolfgang Beitz, *Engineering Design: A Systematic Approach*, Springer, New York, 1988.

Perez-Arriaga, Ignacio J., George C. Verghese, F. L. Pagola, J. L. Sancha, and Fred C. Schweppe, "Developments in Selective Modal Analysis of Small-Signal Stability in Electric Power Systems," *Automatica*, Vol. 26, No. 2, pp. 215-231, 1990.

Procyk, Alexander D., Kim Younkyoo, Einhard Schmidt, Harold N. Fonda, Kwong Chang Chi, Gerald T. Babcock, and David F. Bocian, "Normal Modes of Metallochlorins Revisited: Comparison of Force Fields and Factors Influencing the Vibrational Eigenvectors," *Journal of the American Chemical Society*, Vol. 114, No. 6, pp. 6539-6549, 1992.

Rogers, James L., and Sharon L. Padula, "An Intelligent Advisor for the Design Manager," NASA Technical Memorandum 101558, 1989.

Rosenthal, Stephen R., *Effective Product Design and Development : How to Cut Lead Time and Increase Customer Satisfaction*, Business One Irwin, Homewood, IL, 1992.

Smith, Preston G., and Donald G. Reinertsen, *Developing Products in Half the Time*, Van Nostrand Reinhold, New York, 1991.

Smith, Robert P., "Development and Verification of Engineering Design Iteration Models," Ph.D. Thesis, MIT Sloan School of Management, August 1992.

Smith, Robert P., "Managing Risk through Reordering Decisions in Engineering Design," ASME Design Theory and Methodology Conference, Boston, pp. 585-591, September 1995.

Smith, Robert P., and Steven D. Eppinger, "A Predictive Model of Sequential Iteration in Engineering Design," MIT Sloan School of Management Working Paper #3160, revised April 1994, forthcoming in *Management Science*.

Smith, Robert P., and Steven D. Eppinger, "Deciding between Sequential and Parallel Tasks in Engineering Design," MIT Sloan School of Management Working Paper #3858, October 1995.

Steward, Donald V., "The Design Structure System: A Method for Managing the Design of Complex Systems," *IEEE Transactions on Engineering Management*, Vol. EM-28, No. 3, pp. 71-74, 1981.

Suh, Nam P., *The Principles of Design*, Oxford University Press, New York, 1990.

Strang, Gilbert, *Linear Algebra and its Applications*, Second Edition, Harcourt Brace Jovanovich, New York, 1980.

Taylor, Bernard W., and Lawrence J. Moore, "R&D Project Planning with Q-GERT Network Modeling," *Management Science*, Vol. 26, No. 1, pp. 44-59, 1980.

Throne, Robert R., and Lorraine G. Olson, "A Generalized Eigensystem Approach to the Inverse Problem of Electrocardiography," *IEEE Transactions on Biomedical Engineering*, Vol. 41, No. 6, pp. 592-600, 1994.

Tirion, M. M., and D. ben Avraham, "Normal Mode Analysis of G-actin," *Journal of Molecular Biology*, Vol. 230, No. 1, pp. 186-195, 1993.

Tjandra, Primanata, "Observing Iteration in Engineering Design," Masters Thesis, Industrial Engineering, University of Washington, August 1995.

Tromp, Jeroen, "Support for Anisotropy of the Earth's Inner Core from Free Oscillations," *Nature*, Vol. 366, No. 6456, pp. 678-681, 1993.

von Hippel, Eric, "Task Partitioning: An Innovation Process Variable," *Research Policy*, Vol. 19, pp. 407-418, 1990.

Weber, Jill E., and Christopher S. Campbell, "Breeding System of a Hybrid between a Sexual and an Apomictic Species of Amelanchier, shadbush (Rosaceae, Maloideae)," *American Journal of Botany*, Vol. 76, No. 3, pp. 341-347, 1989.

Wheelwright, Steven C., and Kim B. Clark, *Revolutionizing Product Development: Quantum Leaps in Speed, Efficiency, and Quality*, Free Press, New York, 1992.

Whitney, Daniel E., "Designing the Design Process," *Research in Engineering Design*, Vol. 2, pp. 3-13, 1990.

Appendix A

These appendices contain two extensions to the work transformation matrix model. It is shown here that the two extensions 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 term this a control rule, since this is a work-load policy. We can generalize the control rule. Instead of doing all of the work in every stage, we do a proportion ρ of all work on every task in each stage. The work which is not attempted during the current stage remains to be completed in future stages. Work which is attempted creates work for other tasks as in the original model. The new control rule becomes

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

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

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

We can find the eigenvectors and eigenvalues of the matrix A^* (assuming that the eigenvector matrix is invertible):

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

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

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

Appendix B

The second extension treats time in a more explicit manner. It is shown here that this 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.

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 new work time transformation matrix is written compactly as

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

Repeating the analysis done for the original system, the total work time vector can be found as

$$U^{\dagger} = WS(I - \Lambda)^{-1} S^{-1} W^{-1} u^{\dagger}_{0}$$

Substituting for the initial work time vector,

$$U^{\dagger} = WS(I - \Lambda)^{-1} S^{-1} W^{-1} W u_0$$

which reduces to

$$U^{\dagger} = WU$$

This is the expression originally given for weighting the total work vector by the task times.