Cost Estimation of Functional and Physical Changes Made to Complex Systems

by

Peter Nicholas Jeziorek

B.S. Mechanical Engineering
University of California, Los Angeles, 2003

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN MECHANICAL ENGINEERING
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

FEBRUARY 2005

Signature of Author: _____
Department of Mechanical Engineering
January 14, 2005

Certified by: _____
Nam P. Suh
Ralph E & Eloise F Cross Professor of Mechanical Engineering
Thesis Supervisor

Accepted by: _____
Lallit Anand
Professor of Mechanical Engineering
Chairman, Committee for Graduate Students

Cost Estimation of Functional and Physical Changes Made to Complex Systems

by

Peter Nicholas Jeziorek

Submitted to the Department of Mechanical Engineering
On January 14, 2005 in Partial Fulfillment of the
Requirements for the Degree of Master of Science in
Mechanical Engineering

Abstract

Current cost estimation practices rely on statistically relating physical parameters of a system to historical cost data. Unfortunately, this method is unable to effectively communicate the increasing complexity of system design to cost data. Additionally, current cost estimation techniques have had a historical inability to produce credible and explainable results. It is often considered to be a "black art" with the recurring question: "Where did that number come from?" This thesis systematically links design and cost information together, and demonstrates the utility of that link by estimating the impact of functional and physical design changes on the life-cycle cost and determining key cost drivers. The ability to quickly estimate the cost impact of design changes is important for decision makers and serves as a medium of communication between customers and developers.

Credible estimation is gained by intimately linking the axiomatic design framework to the already existing costing unit (or component) domain and providing design traceability.

Development cost is predicted by determining the functional requirements (FRs) affected by a change in customer needs or constraints, then by determining the propagation of that change from FRs to design parameters (DPs) to costing units. The list of affected components and the magnitude of the impact on each component is found and then used to determine through a parallel iteration process model how much development labor will be necessary to implement those changes. The labor is directly related to development costs.

A formal method to designing operations using axiomatic design is presented in this thesis. Operations exist due to the time-variant combinatorial complexity of FRs. Operations implement reinitialization procedures in order to maximize the probability of success of FRs. This provides the way that axiomatic design can derive operations and the related cost parameters. This information could then be plugged into the cost impact model of a design change to determine the list of affected operations. A new method of estimating the change in cost parameters due to a design change will be the focus of future research.

Two main forms of key cost drivers are identified: the most expensive FRs and design iteration. A method of mapping estimates from the costing unit domain to the FR-DP map is suggested in order to cost out FRs. Design iteration as a key cost driver can be seen from two points of view. Axiomatic design identifies small design ranges, coupling and imaginary complexity as contributors to cost. Design structure matrices identify the most iterative set of tasks in the development process and offer procedures to reduce or speed up the iteration.

Thesis Supervisor: Nam P. Suh
Title: Ralph E & Eloise F Cross Professor of Mechanical Engineering

Acknowledgements

The day I came in to interview for this project, I had absolutely no knowledge of axiomatic design whatsoever. Now after almost nine months, I am able to tackle such difficult design problems such as robustness and complexity. I would not have been able to grow as fast as I had without the help of Professor Nam P. Suh and Taesik Lee.

I'd like to thank Professor Suh for overseeing this project. He has always been a strong supporter of the project, and stood by the cost estimation method that I developed along with Taesik. His style of management has helped me to become an independent thinker and researcher. I am not just regurgitating the ideas of others, but creating new ones. With just one short line from Professor Suh like "Why don't you think about this a little more" or "That's it, huh?" I will add another ten pages to my thesis!

I'd like to thank Taesik Lee for all the discussions we've had concerning axiomatic design and even various matters of life. Often I wouldn't even know what I was trying to say until I tried conveying it to Taesik. He would sit there silent for five minutes and then give some very powerful insight. Without Taesik, this project would have gone nowhere fast. Instead, we hit the ground running and though we slowed down midway, we will be ready to pick up the pace once again to finish the race in first place.

I'd like to thank Lockheed Martin for their support on this project: Robert Ford, Ray Damaso, Chip Woods, Rich Freeman, and Gregory Kerhl. Constant dialogue between MIT and Lockheed has pushed this project forward, making it a very practical and cutting-edge cost estimation tool. I'd like to especially thank Chip Woods for all his correspondence and feedback.

I'd like to thank all the people at Axiomatic Design Solutions Inc. (ADSI): Matt Pallaver, Christian Arangio, and Sung Hee Do. I appreciate all the discussions and the endless number of questions that you guys have.

Thank you Steven D. Eppinger for the presentation you made at ADSI and the discussion that we had thereafter. Your insight and work have been indispensable.

I'd finally like to thank all my lab mates Hrishikesh Deo, Beto Peliks, and AJ Schrauth, for their constant cheer and rowdiness brought to the environment. Without them I would have been done several months ago, but it wouldn't have been half as fun. Thanks Hrishikesh for all your in-depth discussion about Axiomatic Design and Mechanical Engineering. I can always count on you to know what I'm talking about within a few minutes and have good feedback.

Thank you mom and dad. Thank you for your overwhelming support in whatever I do. I respect and love both of you. You are both champion parents and are capable of parenting even the most challenging sons: Alek and I!

Finally I'd like to thank Victoria Tai for all her love and support. She is main the reason I am here at MIT today, and is the reason I am continuing on for a Ph.D. Vicky instantly realized my potential, when I could not. She is an inspiration and an amazing person to spend all my time with. Thank you, Vicky!

I've learned so much, even about the process of learning itself. Learning floats vague ideas somewhere in the depths of your mind. Writing down the idea makes you realize all the problems. You address those problems with countless scribbles and thrown away scraps of paper. Finally, you present the finalized, internalized idea a hundred times. I hope to never stop learning. I hope to never stop growing.

Table of Contents

Chapter 1

Introduction

One of the largest difficulties in developing complex systems is managing and estimating the cost. Life-cycle cost is a measure of the total cost of a system in each phase of its existence: development, production and operation. By knowing the life-cycle cost of a product one can make a decision on whether to proceed with the development of the system or not. This knowledge of the life-cycle cost is invaluable, and can determine the future success or failure of a system. The following examples illustrate the current need for accurate life-cycle cost estimates in complex systems.

The Need for Cost Estimation

In early 1969, Ingalls Shipbuilding Company, by far the largest shipyard in the world, received a contract from the United States Navy to build nine amphibious assault ships (LHAs) for a firm fixed price. The LHA is 20 stories high and the length of three football fields and is capable of deploying 2000 fully-equipped Marines and 200 combat vehicles via landing craft and 30 large helicopters. By the Navy's standards, the LHA was "the largest, fastest and most versatile vessel in the history of American amphibious warfare." In the mid 1970s, Ingalls received another contract from the Navy to produce 30 DD963 Spruance-class destroyers. Ingalls more than doubled its workforce to take on both programs. At the onset of each project, the Navy had only given Ingalls performance specifications of each ship. As a result, at each phase of Ingalls' development process, the Navy interfered by suggesting hundreds of design changes that caused an increase in the amount of work needed to be accomplished, and whose ripple effects seriously slowed down the project. By 1977 Ingalls had filed against the Navy for over $2.7 billion in unsettled claims. What was the mechanism that caused all the delays and cost overrun? How could the cost have been properly estimated? This situation could have been avoided if the Navy and Ingalls Shipbuilding Company had been at an understanding and agreement as to the cost impact of the changes being made.[1]

In 1993, the Big Dig, hailed as the most complex construction project ever attempted in the United States, was a grand plan to revitalize a traffic-plagued Boston. An elevated highway opened in 1959, called the central artery, ran through the heart of Boston, comfortably holding 75,000 passengers per day. By the early 1990s, the highway was squeezing 200,000 passengers per day with 10 hours of standstill traffic. The Big Dig was initiated in order to alleviate the congestion. The concept, originally estimated to cost $3.4 billion, was to replace the central artery with an underground 8-10 lane highway that would culminate in a 14 lane highway and two bridges that would cross the Charles river. In 2004, though construction has nearly finished, the cost of the Big Dig has soared close to $15 billion, repeatedly facing unexpected cost overruns and scrutiny from the Massachusetts residents and state government, as well as the federal government. Why were there so many unpredictable cost overruns? If the decision-makers had known of the actual life-cycle cost of the system, they might have sought cheaper and better alternatives or they might have tried to reduce the costs considerably.[2]

In the mid 1970s, the space shuttle program, the most complex engineered system ever attempted at that time, was conceived to be a replacement of expendable launch vehicles as a cheaper alternative. Today, the cost per unit weight for the shuttle is far higher than expendable launch vehicles. During the 1990's, when comparing other heavy launch vehicles price per pound,

[1] See reference [8].
[2] See references [9] and [10].

the space shuttle tops the list at $4,729 per pound. Other expendable heavy launch vehicles include European Ariane 5G at $4,162 per pound, Chinese Long March 3B at $2,003 per pound, the Russian Proton at $1,953 per pound, and the Ukrainian Zenit 2 at $1,404 per pound. The only advantage of the space shuttle is that it can transport 20,000 lbs more than any of the competitors is manned. The initial intent of the space shuttle program to become a cheaper replacement has been lost. The cost to operate and maintain the space shuttle has proven to be more of a cost penalty than a cost savings. Why couldn't they properly predict the cost of operation far down the line?[3]

On January 14th, 2004, President George W. Bush announced a new and bold vision for NASA. Four goals were outlined by the President:[4]

1. Complete the International Space Station by 2010.
2. Develop and Produce a Crew Exploration Vehicle (CEV) for testing by 2008 and conduct the first manned flight with the CEV by 2014.
3. Return to the moon by 2020, as the launching point for missions beyond.
4. Manned space flight to Mars.

The one thing on everyone's mind is how much will these items cost? Are we willing to pay that much? The current state of cost estimation tells us, that without action, we will not surprisingly be in for huge cost overruns and disappointments. The CEV and continued space exploration systems promise to be the most complex system ever developed and the most costly. We should take care as to develop credible methods of cost estimation that will fully account for the life-cycle cost.

The Current State of Cost Estimation

The root of the problem of cost estimation is that it is part art and part science. There is a degree of subjectivity in estimation. This means that each estimate and methodology can be and is argued over and misunderstood. Parametric estimating methodologies are by far the most prevalently used. Parametric estimates utilize statistical relationships between historical costs and other project variables such as system physical or performance characteristics, contractor output measures, manpower loading, and weight. However, historical costs cannot always predict future costs, since new problems of increasingly larger magnitude continue to surface. The system variables, too, can be very subjective.

Another problem is the "throw it over the wall" behavior between design engineers and cost estimators. The term "throw it over the wall" was made famous over the past few decades by design engineers who would design a product and then give it to manufacturing engineers to "just make it." Frustrated manufacturing engineers would find many faults in the way the product was designed that made it prohibitively expensive to manufacture. As a result, manufacturing engineers and design engineers would enter into many quarrels. Now new methods of designing, such as design for manufacturing, have emerged that ensure that design and manufacturing work hand-in-hand to produce an effective product that is capable of being fabricated. The same situation is occurring today with cost estimators. As long as cost estimators have some type of number to plug into their model, they are happy. The design engineers really have no care for the overall cost estimate; only caring to design the most cost effective and high performance system possible. Cost estimates are too loosely tied to the actual design of a system and the scope of the cost estimation is not well understood. Design engineers are simply throwing numbers over the wall to satisfy the cost estimator. These numbers are not effectively communicating the actual mechanism that is causing cost overrun, and failed estimations.

---

[3] See references [12] and [17].
[4] See reference [11].

In the case of the Ingalls Shipbuilding company, cost overruns occurred due to excessive design changes made by the Navy. Therefore, a quick way of assessing the cost impact of a change to a design must be developed. With such a tool conflicts could be avoided or attended to before any money has been spent and better decisions can be made.

Finally, key cost drivers must be easily identifiable and quantifiable. Key cost drivers are the most significant contributors to the life-cycle cost of the system. By quickly identifying key cost drivers, one can devise strategies on how to minimize the cost impact of these cost-increasing mechanisms.

New ways of estimating cost should be pursued in order to grapple with the new challenges presented to us. These new methods should be concerned with the problems of subjectivity and the lack of communication of the design to the cost estimate. It should provide a quick way of determining the impact of a design change. Finally, it should be able to quickly identify the key cost drivers of a system. The results should be easily communicable and easily understood.

Thesis Objective

With this setting as the background, this thesis is a step forward in the development of an accurate and effective cost estimation model for complex systems. The objective of this model is to facilitate and improve the task of cost engineering to aid in decision making. The model will enhance the credibility of cost estimations and increase utility of the cost information. In particular, this model will accomplish the following tasks:

1. Enhance the credibility of cost estimation by creating the cost model based an Axiomatic Design FR/DP map
2. Quickly estimate the cost impact of changes introduced to a system
3. Identify key cost drivers

This is done by systematically linking three branches of information: system architecture map (FR/DP map), costing-unit interaction model, and process model.

Cost Background and Terminology

It will serve well to first introduce some terms and background information that will be consistently used throughout this thesis.

What is cost? Who is responsible for this cost? Life-cycle cost is a useful all encompassing definition. It is the total cost incurred from the initial development of a system until withdrawal and disposal, and it is the ultimate goal of the cost engineering effort. It includes all the costs due to development, production and operation. Development cost is paid by the developer of the system and typically includes design, testing, and evaluation. Production cost is paid by the producer, who is often the developer or contracted by the developer. This cost includes the cost to manufacture a number of components, including labor, materials, and facility costs; but it does not include the development of those manufacturing systems, which is accounted for in the development cost. Operation cost is paid by the user of the end product. This cost can include the cost of maintenance, energy, time, and disposal. In mathematical lingo, the life-cycle cost is the sum of the development, production and operation costs, seen in Equation 1.

$$\text{Life Cycle Cost} = \$\text{Development} + \$\text{Production} + \$\text{Operation}$$

Equation 1

To further illuminate what development, production and operation means, consider the following example of the life-cycle of an airplane:

1. Development

A company first determines what type of market exists for the airplane. They may spend time researching the current and future trends in passenger flow, including average time and distance flown, how often a person flies, current hot locations, and current Federal Transportation Administration (FTA) regulations. They then begin to lay out requirements in order to meet the customer needs that they want to fulfill. During the conceptual design, these requirements after some work become actual engineering parameters. Engineers diligently work out solutions, exchange information and then iterate, until finally a flyable aircraft that is able to be manufactured is designed. Note that iteration is caused by two phenomena: (1) the exchange of information between people working on dependent tasks, and (2) the correct sequence of tasks is not known before beginning work. Prototypes may be built that test and validate the aerodynamics, structural stability, and avionics. If any faults are discovered, engineers re-iterate and introduce fixes to the design. Meanwhile, the developers search for contractors who can provide certain parts required to produce the system. The company develops the means of producing the desired number of aircrafts. Also, marketing directors attempt to sell the finished product with airlines and foreign nations. A final prototype, which looks like the real airplane, is built and tested before released to the public. Now with many customers waiting for orders, the company is ready to produce the desired lot of airplanes. The cost of development included everything up to this point from the very initial research considerations to the last test flight. Development costs included both labor and material costs accrued during the development phase.

$$\$Development = \$Labor + \$Material$$

Equation 2

2. Production

The company produces a lot of seventy-five airplanes to satisfy the orders that it has received, and promises delivery by a certain date. The cost of production then is associated with running the already existing manufacturing line. Some improvements may be made to the manufacturing line at this time that improve productivity or lower the cost per part. This cost should really be considered development costs. Maintenance, labor, material, and facility costs are included in the production cost.

$$\$Production = \$Labor + \$Material + \$Facility + \$Maintenance$$

Equation 3

3. Operation

An airline company made an order for 10 airplanes. Before they received the airplanes, they spent several months negotiating deals with airports to buy gates and also began hiring new pilots, stewardesses and mechanics. They ran new advertising campaigns, outlining their new flights and prices. The airline prepares itself to use all 10 airplanes to maximize the profit that it can gain from its operation. Finally, once the airplanes are delivered to their respective locations, the airplanes enter normal operating conditions with several flights a day. It maintains a regular schedule of maintenance, with regular tests of its engines, wings and avionics. Finally, after a couple decades of service, the airline company decides that it is more cost-effective to buy a new airplane than maintain the one it is currently operating. As a result, the airline sells, stores, or disposes of the airplane. The cost of operation is the cost of all these activities.

$$\$Operation = \$Labor + \$Consumables + \$Facilities + \$Maintenance + \$Disposal$$

An interesting point to note is that life-cycle cost does not include the benefits gained during the operation of the system, though this is extremely important to the system's marketability. The buyer of the system wants to know exactly how much they will have to pay and what will be the benefits of its use. Life-cycle cost is only concerned with the cost and not the benefits of operation, but life-cycle cost estimates are used in conjunction with the benefits of operation in order to make a product sellable.

Life-cycle cost is also used by the developer to determine how much profit it can gain by continuing with development and production. Making a change to the production phase may make parts cheaper to produce, but it may also reduce the quality of the system during operation. One could spend more money in development, in order to make operations cheaper, and thus make the product more sellable to the customer. These types of games are played by the decision-makers among the developers. In order to make the best decisions possible, they need to be able to see the cost impact of their decision at each stage of the life-cycle.

Summary

Life-cycle cost is the most important measure of the cost during each phase of existence. The three phases of the life-cycle are the development, production and operation stages. The development phase includes all design, testing and evaluation activities from the system conception to directly before the manufacturing of the system. The production phase includes all the activities associated with manufacturing the system with the fully developed manufacturing processes in place. The operation phase consists of all activities that the user of the system must perform in order to properly use the system. These activities include maintenance and using the functions. The cost of each phase is the cost of performing these activities, including all resources, like labor, material, and facilities, required to perform those activities. An example of the life-cycle of an airplane was given to aid in drawing the lines between the three phases of the life-cycle. Life-cycle cost estimates serve in aiding decision makers on whether or not to implement a change to a program. Different scenarios can be posed to see what effect a change can have on each phase of the life-cycle cost. Life-cycle cost does not concern itself with the benefits of operating the system. However, both life-cycle cost and the benefits of a system are important in justifying the final price and worth of the system.

Several examples from industry were highlighted to illustrate the current state of cost engineering. In the 1970s, The Ingalls shipbuilding company received two contracts from the Navy to build state-of-the-art warships. Ingalls was originally given performance specifications by the Navy, but as the design progressed, the Navy interfered with hundreds of design changes. Ingalls, as a result, ran over the allotted budget from the Navy and sought legal reparations in order to pay for the loss. If a method existed in which Ingalls and the Navy could evaluate the impact of each change before they were made, then perhaps the Navy would be more careful in making changes and the project would have been within budget. The Big Dig is the largest and most complex construction project ever undertaken in the United States, and is riddled with cost overruns. It was originally forecasted to cost $3.4 billion, but instead cost $15 billion. What was the mechanism that caused these cost overruns and why was it not estimated correctly? Now the United States boldly states its space exploration goals for the 21st century, of using the moon as a launch pad to the rest of the solar system and manned exploration of Mars. This will be the most difficult and challenging engineering project ever attempted, and will likely be the most costly. Before embarking on such a journey, proper cost estimating techniques should be developed that would prevent the problems experienced by Ingalls and the Big Dig and countless other firms that engineer large complex systems.

The problems with current cost estimating practices are the following:

1. Cost estimation is a "black art," consisting of one part science and one part art. The art of an estimate comes from a certain level of subjectivity in the estimate.

2. Cost estimation today is based on historical data linked to physical parameters of a design. On one hand, historical cost data that reflects past problems in designing systems cannot always predict the cost of future problems. On the other hand, the physical parameters that attempt to describe a design do not embody the actual design and have a degree of subjectivity.
3. As a result, a "throw it over the wall" syndrome that was associated with design and manufacturing engineers in the past is now present among design engineering and cost estimators. The cost estimator only cares to receive the number he needs to plug into his model, while the design engineer only cares to design the best performing and cost-effective system possible. The communication that currently exists is not sufficient in communicating the foreseeable and unforeseeable problems that can exist in the design. This results in the cost estimates not accurately reflecting the problems occurring in the design.
4. Key cost drivers must be quickly and easily identified for the purpose of cost minimization.
5. The results must be easily understood and justified.

With these problems evident in current cost estimation practices, the goal of this thesis is to:

- Enhance the credibility of cost estimation by creating the cost model based on the FR/DP map
- Quickly estimate the cost impact of changes introduced to a system
- Identify key cost drivers

These goals are accomplished by using the Axiomatic Design Framework along with a Process-Based Model. The method is expounded upon in subsequent chapters.

Chapter 2

Enhancing the Credibility of Cost Estimation with Axiomatic Design

Enhancing the credibility of cost estimation is the first goal of this model. Recall from Chapter 1 that there exists an information exchange between designers and cost estimators. Currently, this exchange takes the form of the passing of different types of physical variables that loosely describe the design and relate to historical cost data. Unfortunately, this communication is not adequate enough to transfer the design information. Key mechanisms such as design coupling and iteration are not represented clearly in the cost domain in a way that reflects the true design. Additionally, the cost information solely resides in the physical domain. Only parameters such as power, torque, weight, number of gear teeth, etcetera are collected and then translated into costs. Customer needs, functional requirements (FRs), and design parameters (DPs) are in no way currently connected to the cost information. This chapter outlines how these vital aspects of the design are joined to the cost domain. The key element is the Axiomatic design framework with the addition of the Costing Unit domain. Additionally, this chapter demonstrates how axiomatic design improves the traceability of a design. Cost estimation that is based on a traceable design will be more credible.

An Introduction to Axiomatic Design

Axiomatic design is a design methodology that is useful in guiding the design process. It was developed in order to formalize the oftentimes haphazard procedures of design commonly practiced in industry. Axiomatic design states that a design develops from customer needs into FRs. Each FR must be satisfied by a single DP. The relationship between each FR and DP is captured by the design matrix. The independence axiom states that each FR must be independent of the other FRs. The information axiom states that the information content of a design should be minimized. By using these two axioms along with a rigorous process for design, one can design a system that best satisfies all the FRs, without unnecessary iteration.

A product is developed solely to satisfy a customer's needs. The customer is willing to pay for his/her satisfaction, so that the developer can earn a profit. Therefore, the start of every design is in the customer needs (CN) domain. Depending on whom the customer is this process could be quite involving or easy. In the case of designers in the aerospace industry, they may only have one customer, such as NASA or the Department of Defense. This one customer may clearly states its needs. In other cases, when there are many customers, marketing research must be performed in order to gain insight into what the customer's needs are.

From the CN domain, a designer must unravel the minimum set of independent FRs required to meet each CN. Thus, the designer must create a map from the CN domain to the FR domain. The first design axiom, the Independence Axiom, states that the designer should maintain the independence of the FRs. Similarly, as each FR is defined, a design parameter (DP) must be defined in order to satisfy that functional requirement. That design parameter may have certain functional requirements of its own, therefore, requiring further breakdown. This process is called zigzagging because the design progresses by jumping between both the FR and DP domains, as seen in Figure 1. FRs can also be subject to certain constraints that limit the range of possibilities of that FR. Constraints can be either introduced into the system by the designer, or can be inherent, like the laws of physics.

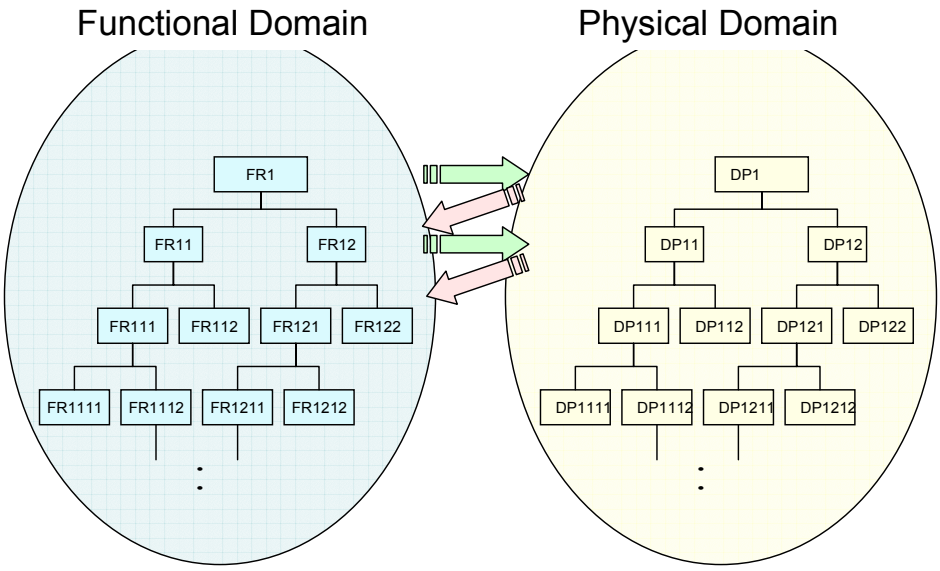

Figure 1. Axiomatic Design process creates a hierarchical description of a system by zigzagging between the functional and physical domains.

The relationship between FRs and DPs is represented by the design matrix. Ideally, each DP should only affect a single FR. The design matrix of such a system is said to be uncoupled (Figure 2a). Having one DP affect multiple FRs causes many problems. The design matrix of such a system is said to be decoupled (Figure 2b) or coupled (Figure 2c). Notice in Figure 2b that if DP3 is changed, then it affects FR1, FR2, and FR3. A decoupled design must be performed in a certain sequence in order to prevent iteration. A coupled design requires multiple iterations in order to satisfy the functional requirements. The X at ijth position of the matrix in the design matrix denotes that a relationship between the ith FR and the jth DP exists. Also, A_{ij} can be a variable relating FR_i and DP_j.

$$\begin{Bmatrix} FR_1 \\ FR_2 \\ FR_3 \end{Bmatrix} = \begin{pmatrix} X & 0 & 0 \\ 0 & X & 0 \\ 0 & 0 & X \end{pmatrix} \begin{Bmatrix} DP_1 \\ DP_2 \\ DP_3 \end{Bmatrix} \qquad \begin{Bmatrix} FR_1 \\ FR_2 \\ FR_3 \end{Bmatrix} = \begin{pmatrix} X & 0 & 0 \\ X & X & 0 \\ X & X & X \end{pmatrix} \begin{Bmatrix} DP_1 \\ DP_2 \\ DP_3 \end{Bmatrix} \qquad \begin{Bmatrix} FR_1 \\ FR_2 \\ FR_3 \end{Bmatrix} = \begin{pmatrix} X & X & X \\ X & X & X \\ X & X & X \end{pmatrix} \begin{Bmatrix} DP_1 \\ DP_2 \\ DP_3 \end{Bmatrix}$$

(a) uncoupled (b) decoupled (c) coupled

Figure 2. Varying degrees of coupling in design matrices. (X's mark a relationship, 0's mark no relationship)

The complexity of a design is measured in the uncertainty of the DPs ability to satisfy the FRs. In a time-invariant design, the uncertainty is the probability that an FR will be successfully fulfilled. This probability is captured by the overlapping of the design range and the system range. The design range is the acceptable range of values in which an FR can reside. The system range is the range of values in which the FR actually resides. For example, suppose that a FR is satisfied on the design range of 0.125. +/- 0.03125 and the FR actually remains uniformly distributed on the system range of 0.109375 +/- 0.015625. This system range is entirely within the design range, and therefore that FR has a 100% probability of success. Suppose that, instead, the FR's system range was 0.25 +/- 0.125 with uniform probability. This determines that the probability of success of that DP satisfying that FR is 0.03125/0.25 or 12.5%.

In general, the design range is the acceptable range of values that an FR can take, and the system range is the actual range of values that the FR takes, typically represented by a probability

density function, seen in Figure 3. The probability of success is the area underneath the curve of the probability density function, f_{FR}, of the FR on the design range (DR). This relationship is captured in Equation 4.

$$\mathrm{P}(\mathrm{FR_i\ Successful}) = \int_{DR} f_{FR_i}(x)dx$$

Equation 4

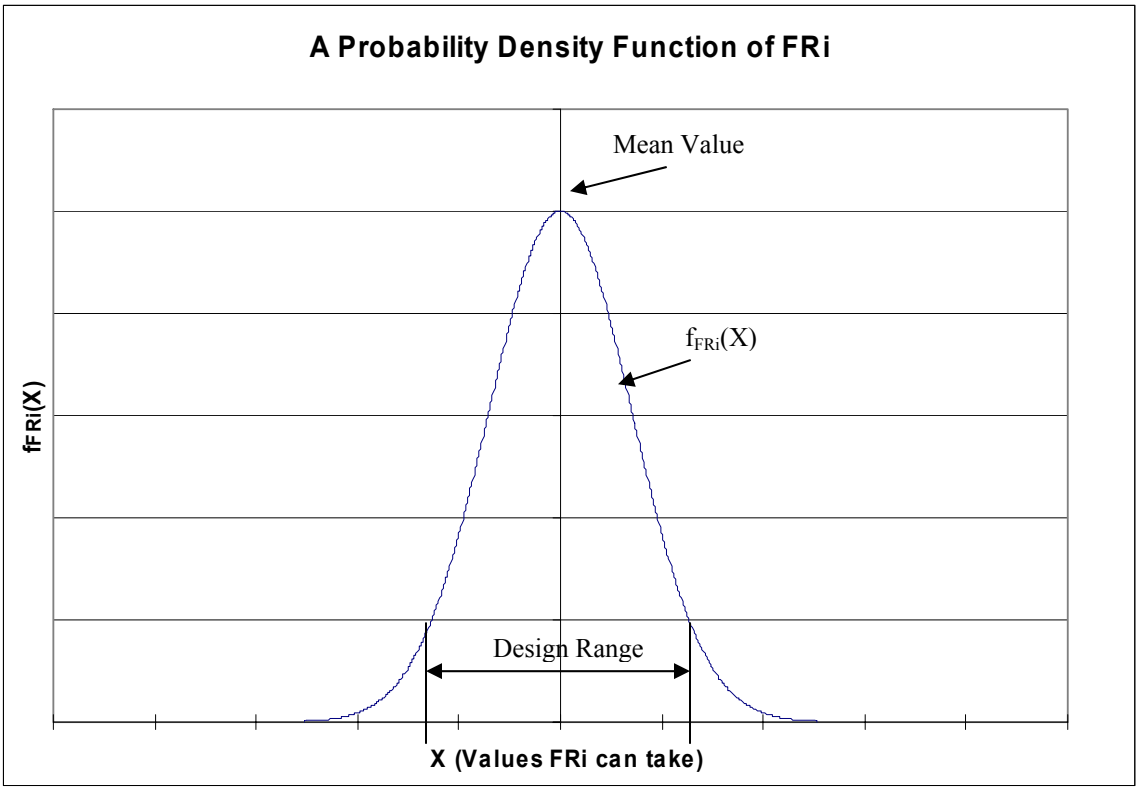

Figure 3: Probability Density Function of FR_i

The uncertainty can also be represented in another way, called information. The information of the uncoupled FR_i is defined in terms of the probability P_i of Satisfying FR_i, as seen in Equation 5.[5]

$$I_i = -\log_2 P_i$$

Equation 5

The information axiom states that the designer should minimize the information content of the design. The best design therefore is the one with the least information content, or in other words, the best design is one that has the highest probability of satisfying all the FRs. Information content, therefore, becomes a useful measurement of the complexity of a system, which can also be defined in terms of uncertainty of being able to fulfill all the FRs of the system.

Thus, the design information is completely captured in the Axiomatic Design framework, from the customer needs domain, to the functional and physical domains. The complexity of the design is captured by the uncertainty in fulfilling all the FRs.

[5] For a more in depth description of information see p. 39 of reference [1].

Connecting Design and Cost Information

In order to connect the design description of the system with cost information, another domain is created. This new domain is termed as the *costing unit* domain. The costing units (CUs) are physical entities that represent the actual system. To some degree, it is equivalent to the bill of materials (BOM), list of components, or work breakdown structure (WBS). CUs are not the same as DPs, since DPs can be variables or characteristics that are not necessarily physical parts. For example, a beverage can has 12 FRs and 12 corresponding DPs, but only three CUs – integrated physical components. The design of the main body of the can should satisfy the FRs of containing radial and axial pressure and withstanding impact from a 2m height by modifying DPs like the material, thickness, radius, length, and convex shape of the bottom of the can.[6] In our model, FRs, driven by customer needs, are mapped into DPs, and DPs are mapped into CUs. This chain of information enables us to assess the impact of the FRs on cost. Thus, information that historically has resided in two separate domains is now integrated together, as seen in Figure 4.

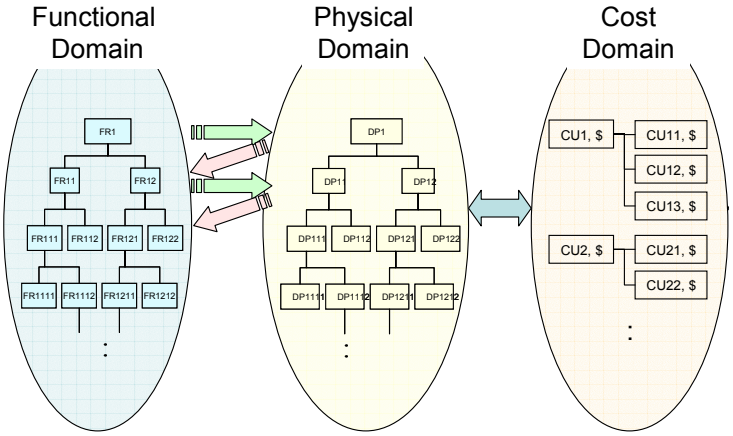

Figure 4: FRs and DPs are connected to CUs, naturally linking design and cost information together.

The FR-DP structure in our model is hierarchical. As the design matures from conceptual solutions to detailed solutions, the FR-DP map develops deeper levels of hierarchical structure. Since the cost information is closely tied to the FR-DP map in our model, it naturally produces a system design cost breakdown from the top-level to leaf-levels. This cost breakdown, along with the hierarchical design description, offers a good way to manage cost information. In the next chapter, one example of the utility of this link will be shown. The framework will be used to track the propagation of a design change to a functional requirement into the cost domain.

Traceability

Traceability is being able to track the relationship between requirements and the solutions that satisfy those requirements. How does Axiomatic design aid in traceability? Refer to Figure 5 and Figure 6. Notice that the design parameter, DP121 affects FR121, FR122, and FR123. Suppose that we need to determine the impact of a requirement change to FR121. In order to satisfy the change in that requirement, DP121 will have to be changed. Because DP121 affects FR122 and FR123, then those requirements may not longer be satisfied after DP121 is changed. This, therefore, implies that DP122 and DP123 must change as well. Note that a change to DP123 will propagate to the lower levels FR1231 and FR1232. Therefore, traceability is obtained by identifying which solutions are likely to change due to a requirement change and then see what

[6] See p. 19 of reference [1].

other requirements may be affected. The design matrix provides an easy way to determine which requirements and design solutions will be affected by a change.

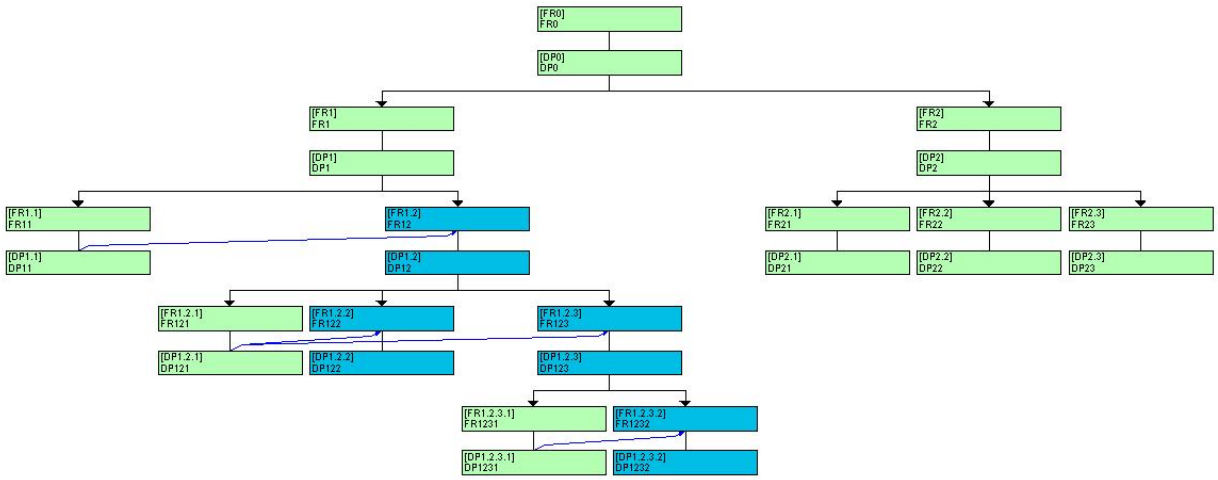

Figure 5: A Requirement and Solution Decomposition

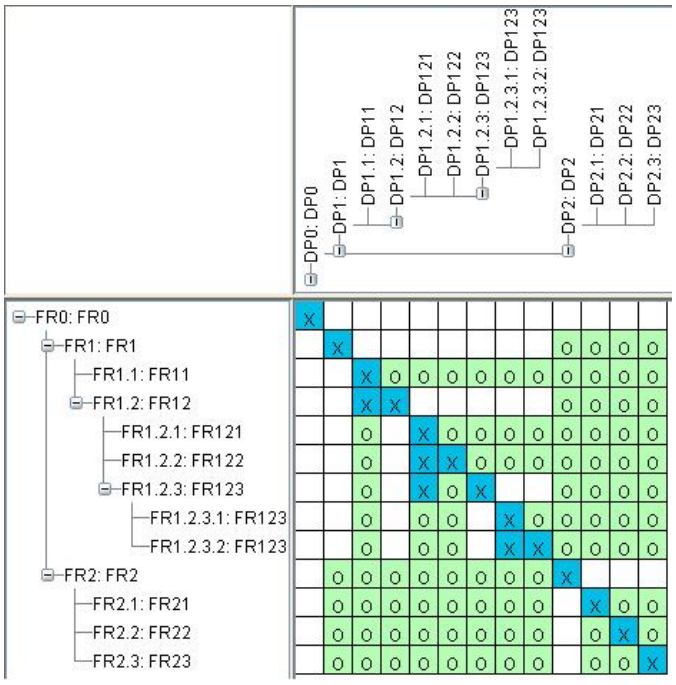

Figure 6: A matrix highlighting the relationship between solutions and requirements.

The current process of capturing traceability in the industry lacks DPs and their relationship to requirements. Instead only requirements and their related hierarchy are captured. Suppose that a design had the same situation as described by Figure 5. Following the industry's process, the requirements hierarchy would look like the one in Figure 7.

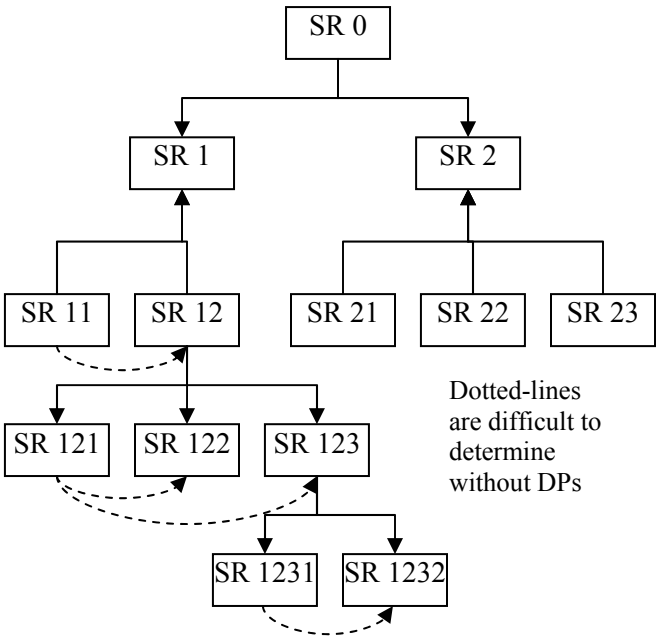

Figure 7: A hierarchial system requirement (SR) decomposition of the same design as in the previous two Figures.

In order to provide an adequate amount of traceability (the same as provided when design parameters are used), SR 11 should link to SR 12, SR 121 should link to SR 122 and 123, and SR 1231 should link to SR 1232. These links are represented by the dotted-lines in Figure 7. This would provide the same result as that from the previous two figures. The way that these links are typically determined is through design parameters. So whether or not design parameters are recorded, the links are established by design parameters. There would be no other way to determine how two seemingly unrelated system requirements are related. It is only by determining an intermediate parameter that affects whether both requirements are satisfied. Design parameters are inherently in the design process, but are typically not recorded and related to functional requirements. Axiomatic design requires the designer to record design parameters and their relation to requirements. Traceability, therefore, is inherent in the Axiomatic design process.

The credibility of a cost estimate is enhanced by traceability because it provides additional information about the design that was not available before. For example, when a design change is introduced, it will become immediately apparent which functional requirements, design parameters, and costing units are involved in that change. This helps to define what the scope of the design change should be. When a design change is introduced, without traceability it is difficult to determine which costing units will be affected. As a result, the actual list of costing units that will be affected by a change may be different from what is identified and, therefore, the scope of the design change could be incorrect. A cost estimate based on an incorrect scope, no matter how accurate the cost estimating tools used, will still be incorrect. The next chapter shows how traceability is used to define the scope of a design change.

Summary

Current cost estimation practices do not adequately take into account all aspects of the design, including customer needs, functional requirements, design parameters, and complexity. They solely rely on physical parameters of a design like weight, power, number of gear teeth, etc to determine cost information. Historical data attempts to fill in the gap by taking into account

past trends in design complexity and iteration. Our model enhances the credibility of cost estimation by linking design and cost information together.

The axiomatic design framework provides a good description of the design. It maps the customer needs into functional requirements (FRs), and then to design parameters (DPs). The relationship between FRs and DPs is represented by the design matrix. The design matrix is essential in revealing the complexity of the design, which is defined as the uncertainty in fulfilling all the FRs. This uncertainty is measured in terms of the overlap of the design and system ranges. The design matrix also uncovers the coupled nature of the design. This coupling can either be eliminated through developing a better design or can be dealt with properly by varying the DPs in the correct sequence or through design iteration. When used correctly, axiomatic design becomes a great tool to aid in the designing process and results in a solid description of the design.

Design and cost information are linked together by linking the FR and DP domains to a new costing unit domain. A costing unit (CU) can be thought of as a component of the Work Breakdown Structure (WBS) and is the typical object of the cost estimation efforts. The link occurs by mapping DPs to the respective CUs. Since the cost information is closely tied to the FR-DP map, it naturally produces a system design cost breakdown from the top-level to leaf-levels. The utility of this link is shown in the following chapter in determining the cost of a design change.

Traceability is being able to track the relationship between requirements and the solutions that satisfy those requirements. The axiomatic design process requires the designer to record all the design solutions and their relation to FRs. By doing so, the designer is inherently increasing the traceability in the design. Traceability aids in defining the scope of a design change. Without traceability, a team of experts must come together and try to identify the requirements and solutions that will be affected by the change. The team of experts can potentially include unaffected requirements and solutions and exclude others that are affected. The result is that the scope of the cost estimate is erroneous. With traceability, it immediately becomes apparent which FRs, DPs and CUs will be affected by the design change. The scope of the design change is clearly and correctly defined, adding credibility to the cost estimate.

Chapter 3

Quickly Estimating the Cost Impact of a Design Change to Development

How does a design change affect the life-cycle cost of a system? First of all, it will help to define what a design change is. A design change is usually implemented by a designer either because the customer needs or constraints have changed, or because the functional requirements (FRs) directly have been changed. Design parameters (DPs) are then modified in order to meet the changed FRs. Also, technical advances or roadblocks may encourage the use of a different DP. The general procedure in determining the cost impact of a design change should start with determining which DPs must change. Then a magnitude of the change to each DP should be determined. This magnitude should then translate into an increase or decrease in life-cycle cost.

This chapter focuses on determining the cost impact of a design change to the development cost. The general method of determining the cost impact of a design change applies here. A set of FRs are identified as changed, thereby requiring certain DPs change. The set of DPs that must change are determined from the design matrix. Now, in order to estimate the development cost, the affected components, or CUs, are identified by a DP-CU, and CU-CU matrix and the time required to complete the design changes for each component is determined using a Task-based model. This procedure takes into account the physical interactions between components, in order to determine how the change propagates through the system. The additional iterations required to make the change are determined by a Task-based model, which quantifies the coupling of tasks. Once the additional time required to finish each component after a design change is determined, the time directly translates into labor costs. Historically, the development labor cost is approximately directly proportional to the total development cost, which includes material and labor costs. The total development cost due to the design change, then, is estimated.

Once the model is constructed, all that is required is the input of several parameters in order to determine the cost impact of a design change. This model once encoded into software, therefore, is an extremely quick and useful tool that aids in the management process as design changes are being considered for implementation. Typically, it takes organizations several weeks to determine what the actual impact of a design change will be to the design and life-cycle cost. Their analysis is not always guaranteed to be complete either. Instead, this model outputs the affected components, how they are affected, and what the cost impact will be. The output from this model also serves as a medium of discussion about design changes for organizations like Ingall's and the US Navy. The implications of this model, therefore, are of interest to not only cost estimators, but to the system-developing organization as a whole.

Identify the Components Affected by a Functional Change

The Axiomatic design framework provides a mapping from customer needs into FRs, or the set of functions that the product must perform in order to satisfy the needs of the customer. FRs are then mapped into DPs, or specific engineering parameters that are varied in order to perform the desired functions.[7] By doing this, a clear connection is established between the customer's needs and the actual product being designed. The relationship between FRs and DPs, as seen in Figure 8, is of special interest. There are three FRs and three DPs that are related to each other by this matrix. FR1 is affected by DP1, FR2 is affected by DP2 and FR3 is affected by DP2 and DP3. Or in other words, if DP1 is changed, FR1 will be affected, if DP2 is changed, FR2 and FR3 will be affected, and if DP3 is changed, FR3 will be affected. Later, this

[7] See Chapter 2, reference [1] and [2] for more detailed information about Axiomatic Design.

information will be used to identify which DPs will be necessary to change in order to satisfy a change to a functional requirement.

	DP1	DP2	DP3
FR1	X		
FR2		X	
FR3		X	X

Figure 8: The FR-DP Relationship

The V-model, shown in Figure 9, illustrates how the design process works. The left-hand part of the V shows the top-down axiomatic design approach, which was described at length in Chapter 2. FRs are developed from customer needs. DPs are then created that can satisfy those FRs. The decomposition process unravels more details of the design. Once this process is completed, all the physical components, or costing units (CUs), must be identified. CUs are the objects of the cost estimation, and the physical artifacts generated from the design. For the beverage can, the three physical components of the can would be the three CUs. Each DP can be embedded in multiple CUs and each CU can embody multiple DPs. The reason one DP can map to several CUs originates from the fact that designers create CUs that are at a lower level than the available DPs. For example, one FR may state "Provide fuel to the engine" and the corresponding DP would be a Fuel Delivery System. This may include CUs such as a fuel tank, fuel pump, fuel lines. This higher level DP would correspond to these three CUs in the absence of further decomposition. After full decomposition, this problem will no longer occur. The relationship between DPs and CUs can be seen in the example in Figure 10. CU1 contains DP1 and DP2; CU2 contains DP1, DP2 and DP3; CU3 contains only DP4. Finally, once all CUs have been identified, these CUs must be integrated into a complete system.

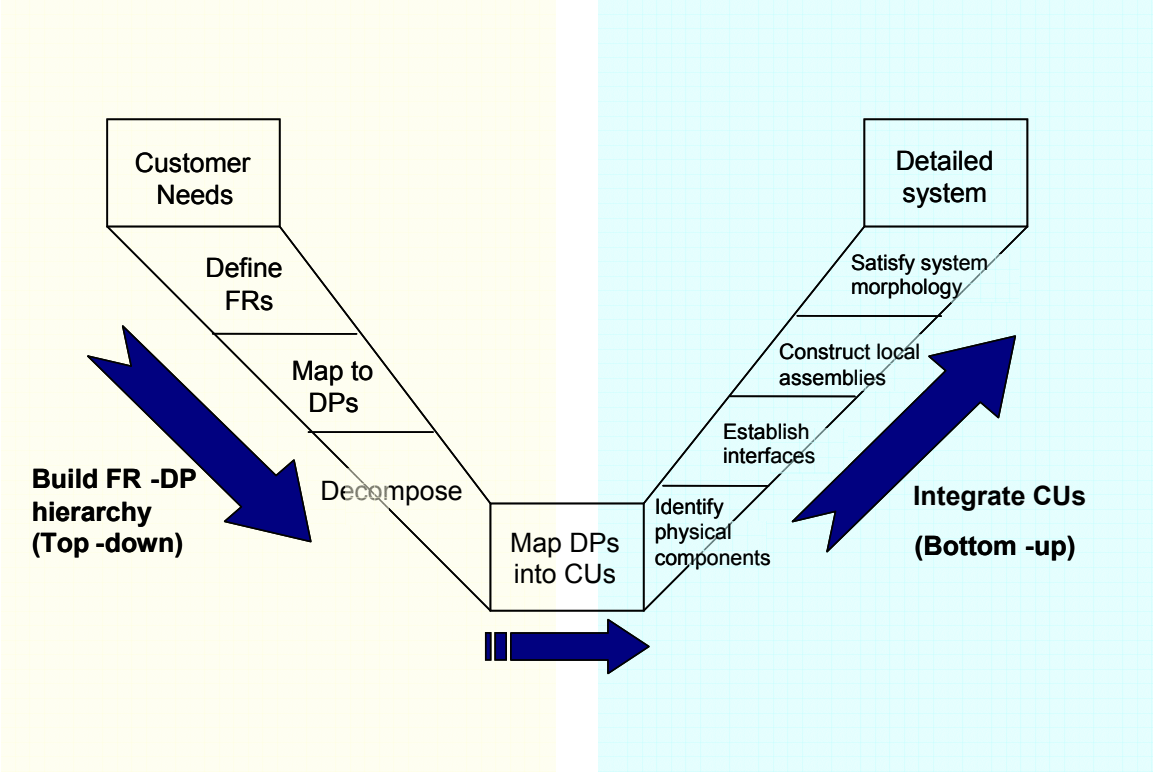

Figure 9: The V-Model illustrates how DPs are integrated into physical entities, or CUs.

	CU1	CU2	CU3
DP1	X	X	
DP2	X	X	
DP3		X	
DP4			X

Figure 10: The DP-CU Relationship

With the FR-DP and DP-CU matrices, we can determine the list of components that are affected by a functional change. Suppose that the following design matrix, in Figure 11, was under consideration for a significant design change. In order to best satisfy the customer needs, a manager decides that FR1 will have to change. Consequently, in order to satisfy FR1, DP1 will also have to change. Because DP1 also affects FR6, DP6 will have to be changed in order to compensate for the change in DP1 and still satisfy FR6. The result of the change to FR1 is that DP1 and DP6 will have to change.

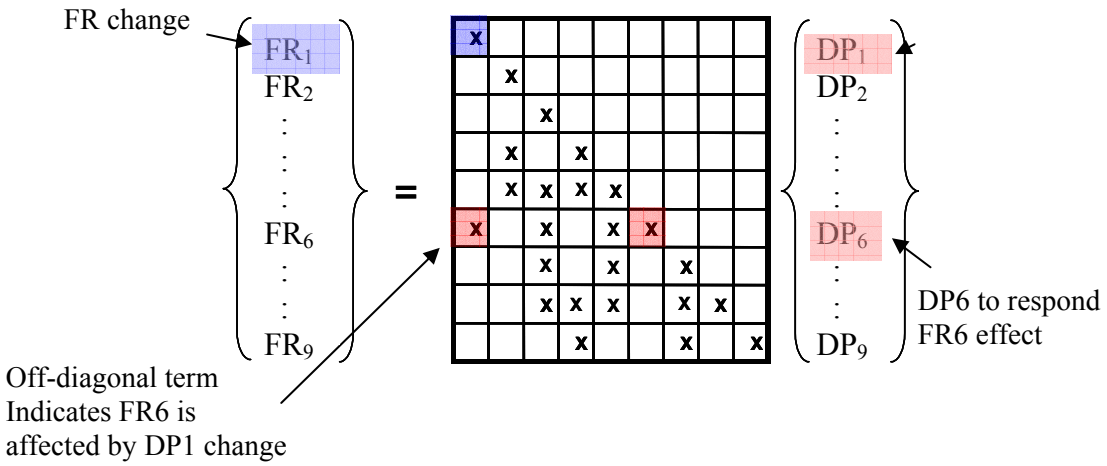

Figure 11: A change to FR1 requires a change in DP1 and DP6

From the DP-CU relationship, seen in Figure 12, we can find the CUs (components) that will be affected by the changes to DP1 and DP6. Reading from left to right and then up, we can identify CU1 and CU3 as the components that will need to be changed as a result of the change to FR1. Thus the output from the DP-CU matrix is a list of components affected by the functional changes.

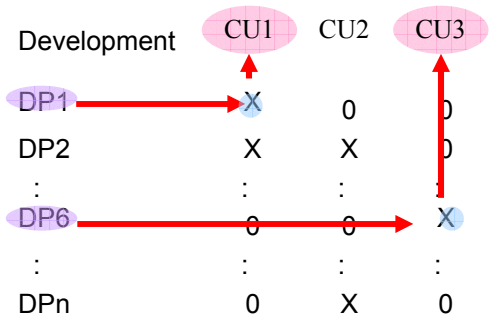

Figure 12: A change in DP1 and DP6 necessitates a change in CU1 and CU3.

This list of affected components only takes into account functional interactions between DPs. Many components interact with each other physically as well as functionally. However, this information is typically not captured by a design matrix. Instead, a new CU-CU matrix was created in order to capture physical interactions between CUs. Imagine a set of components that physically interact with each other, as in Figure 13. Five examples of component attributes that can interact with other components are physical, spatial, thermal, information and electromagnetic. The physical attribute indicates that this component physically integrates with another component, for example, by a mount or tubing. The spatial attribute specifies size or location of the component. The thermal attribute marks the presence of heat exchange or generation. The information attribute is the flow of information into and out of a component. The electromagnetic attribute is the transmission of an electromagnetic signal into or out of a component. A component can interact with its neighbors through any combination these five attributes. For example, In Figure 13, the physical attribute of Component 1 interacts with the physical attribute of Component 3 and the information attribute of Component 1 interacts with the electromagnetic attribute of Component 2. These interactions are two way because of the question that we ask: "Does the information attribute of Component 1 interact with the electromagnetic attribute of Component 2?" "Does the electromagnetic attribute of Component 2 interact with the information attribute of Component 1?" is the same question. The answer is yes for both questions, since they are the same question. Even an attribute within component 1 could interact with another attribute of component 1. In the example in Figure 13, the information and electromagnetic aspects of component 1 interact with each other.

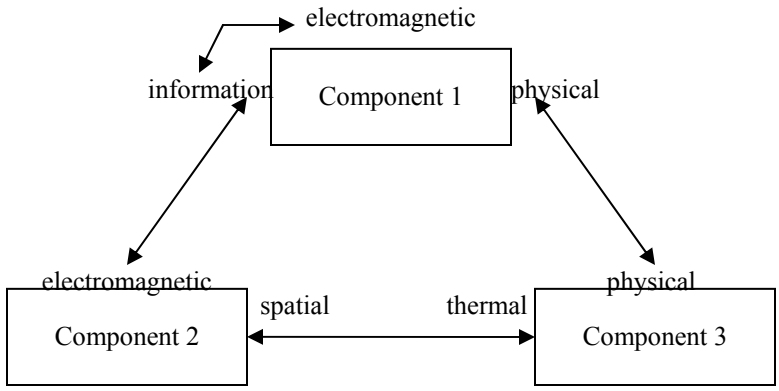

Figure 13: Component Interactions

Unfortunately, diagrams like this become difficult to create and read as the number of components or CUs increases to the hundreds, or even thousands. A CU-CU matrix better captures this information. Essentially, the CU-CU matrix is a collection of two-way pointers that connect the CU attributes. By definition, this matrix is symmetric. The equivalent matrix of Figure 13 can be seen in Figure 14. This matrix is also similar to that of a component design structure matrix.

Costing Unit	CU_1	CU_2	CU_3

	Aspect	physical	information	electromagnetic	spatial	electromagnetic	physical	thermal
CU_1	Physical	X					X	
	information		X	X		X		
	electromagnetic		X	X				
CU_2	Spatial				X			X
	electromagnetic		X			X		
CU_3	Physical	X					X	
	Thermal				X			X

Figure 14: The CU-CU Matrix

By using a CU-CU matrix, the list of affected CUs from a functional change can be refined and expanded. The following example will demonstrate the propagation of changes through interfaces of components. Recall from Figure 12 that CU1 and CU3 had been identified as the affected CUs by a functional change in FR1. Suppose that the interactions between the three CUs were characterized by the CU-CU matrix in Figure 14. Certain attributes of CU1 and CU3 will change to satisfy the FRs that had been changed. Suppose that the information attribute of CU1 and the thermal attribute of CU3 were changed. This type of change would necessitate a change in the electromagnetic attribute of CU1 and the spatial and electromagnetic attributes of CU2. The interface between the information and electromagnetic attributes of CU1 is represented by the two X's at (2,3) and (3,2). Because this interface was identified as necessary to change, both X's are marked. The method of determining the affected interfaces is known as change propagation outlined in Figure 15. All the X's in the column of a changed attribute are marked as necessary to change. For example, following the line from the thermal attribute of CU3 identifies the spatial attribute of CU2 as being affected. The input to the CU-CU matrix is the information attribute of CU1 and thermal attribute of CU3 and the output is the information and electromagnetic attributes of CU1, the spatial and electromagnetic attributes of CU2, and the thermal attribute of CU3. This successfully identifies the complete list of affected CUs and even the attributes and interfaces of the CUs that will be affected by the change. This step can be repeated with the output as input in order to reflect further propagation of the change.

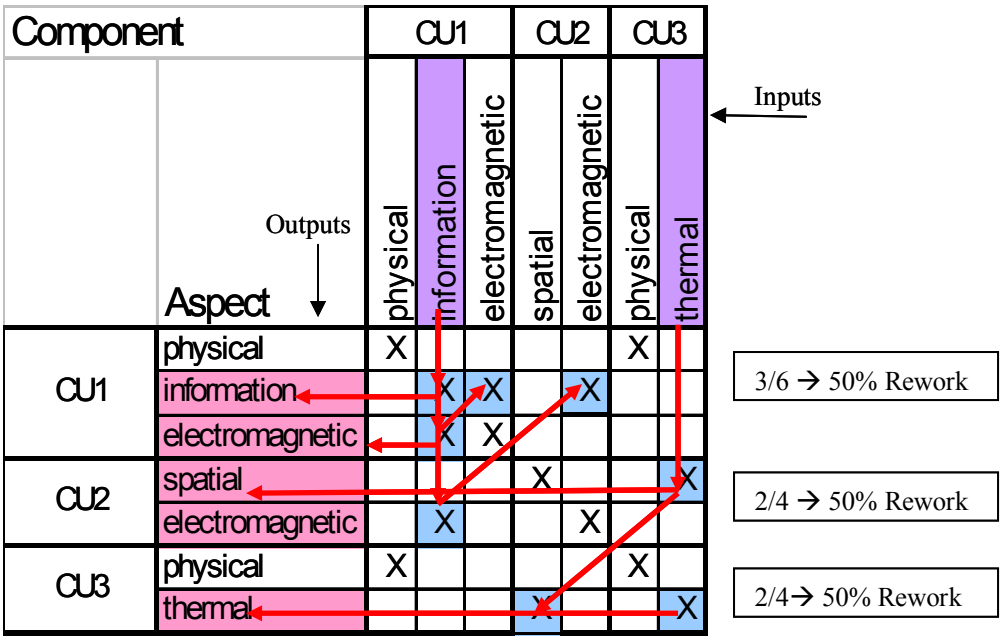

Figure 15: Determining the complete list of CUs affected by a functional change by analyzing the interactions between CUs.

The nature of propagation of a change can be understood better through an example. Suppose that in the middle of the design of a car, it is decided that the size of the trunk must change. The trunk is connected to the frame of the car, and therefore, a change in the size of the trunk could affect the design of the frame. The frame of the car is also connected to the engine bay, and if the frame design changes, then perhaps the engine bay design must also change. If the engine bay design changes, perhaps the engine, intake, headers, or throttle body may have to change. This question could continue until all components had been identified as affected due to a change in the trunk. This is not very useful for cost estimation analysis. Instead, the "nearest neighbor" rule is employed. The nearest neighbor rule states that a change to a component only propagates changes to components that are directly connected to that component. For the CU-CU matrix, this means that the process of change propagation is only performed once.

A measure of the magnitude of a change is the number of affected interfaces or attributes out of the total number of interfaces and attributes. Note that interfaces are indicated by off-diagonal X's, while diagonal X's indicate attributes in a CU-CU matrix. We later use this measurement to determine the amount of rework that must be done to complete the design change. The calculation of percentage rework can be seen in Equation 6.

$$\%\ \text{Rework} = \frac{\#\ \text{of Affected Interfaces or Attributes}}{\text{Total}\ \#\ \text{of Interfaces and Attributes}}$$

Equation 6

Take, for example, CU1 from Figure 15 and calculate the % Rework. There are three attributes and three interfaces, for a total of six interfaces and attributes.[8] The information attribute will have to change and the CU1 Information – CU1 Electromagnetic, and CU1 Information – CU2 Electromagnetic interfaces will have to change, for a total of 3 affected interfaces and attributes. The % Rework is then calculated to be 3/6, or 50%. This indicates that 50% of the CU1 design must be redone, while the other 50% can still be used.

[8] The X's at (3,2) and (2,3) count as one interface within CU1.

Determine the Development Labor Cost

Now that the complete list of CUs has been identified and the amount of rework required for each CU has been calculated, the amount of development time required to implement those changes can be determined. By measuring the impact of a change on the development time of a project, the cost of labor can be determined. This is accomplished using a task based model.[9] Recall from Equation 2 that the total development cost is the sum of the labor and material costs accrued during the development phase. This section is specifically dealing with the labor portion.

The task based model is a record of the interaction between all the processes in a project and a means of calculating the time required to complete each task. There are three different configurations of two tasks A and B, as seen in Figure 16. If task B requires information from task A, the task A must be completed before task B. These two tasks are sequenced in series. If tasks A and B do not require information from each other, then they can be performed in parallel. If tasks A and B require information from each other, then they have to be performed in parallel with iterations involved. We chose to model tasks as being done all at the same time (in parallel), though development typically involves a mixture of the three forms of task sequencing.

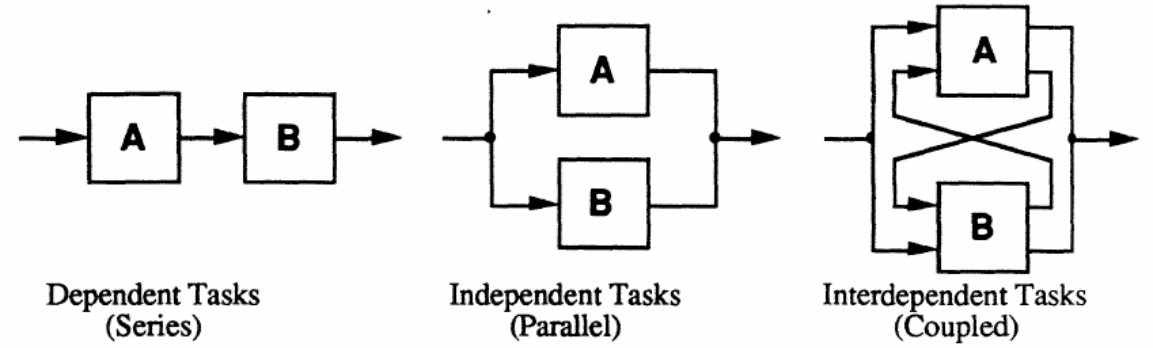

Figure 16: Three possible sequences for two tasks.[10]

Drawing figures aids in the visualization of how tasks interact, but this practice becomes less useful and more cumbersome as the number of tasks increases. The work transformation matrix, WT, is used instead, as seen in Figure 17. This matrix shows that tasks A and B are coupled, interdependent tasks. In a parallel iteration model, tasks run through an iteration, and then exchange information, thereby creating additional work called rework. Both tasks iterate in this way until both tasks are completed. The total time in order to complete each task, therefore, is the sum of the time spent on each iteration. The values in the matrix represent the percentage of rework created at the end of each iteration. The work transformation matrix in Figure 17 indicates that task B creates an additional 50% of rework for task A and that task A creates an additional 30% rework for task B after each exchange of information.

	A	B
A		0.5
B	0.3	

Figure 17: A Work Transformation Matrix, WT, with Two Coupled Tasks

Mathematically, the total time required to complete two tasks in parallel is the following. The initial work vector, u_0, represents the initial amount of work required to complete each task

[9] See reference [3] and [4].
[10] From reference [4]

assuming completely independent tasks. This is represented by a column vector of 1's of length m, where m is the number of tasks.

$$u_0 = \begin{Bmatrix} 1 \\ \vdots \\ \vdots \\ 1 \end{Bmatrix}$$

Equation 7

In order to find the amount of rework required to complete the n^{th} iteration, we multiply u_{n-1} by WT.

$$u_n = WTu_{n-1}$$

Equation 8

For example, to calculate u_1 for the WT matrix in Figure 17, multiply WT and u_0:

$$u_1 = WTu_0 = \begin{bmatrix} 0 & 0.5 \\ 0.3 & 0 \end{bmatrix} \begin{Bmatrix} 1 \\ 1 \end{Bmatrix} = \begin{Bmatrix} 0.5 \\ 0.3 \end{Bmatrix}$$

As n gets large, the work vector should converge to 0. If it doesn't, this means that the tasks are too interdependent and will continue to iterate forever. The total work vector, U, is the sum of all the work vectors.

$$U = \sum_0^N u_n = u_0 + \sum_1^N WTu_{n-1} = \sum_0^N WT^n u_0 = \left(\sum_0^N WT^n \right) u_0$$

Equation 9

The total work vector, U, for the example in Figure 17 is calculated to be $U = \begin{Bmatrix} 1.7647 \\ 1.5294 \end{Bmatrix}$, with the u_{10} converging to the zero vector, a vector in which all its elements are zero.[11] This means that task A will have to do 76.47% more work due to the dependency on task B and that task B will have to do 52.94% more work due to the dependency on task A.

With a task based model, we can calculate the amount of time required to complete each component (CU). Each component has a certain set of interdependent tasks required to complete that component. For a system with thousands of components, it could become tedious to identify all tasks required to complete each component and the interdependencies between tasks. One solution is to assume that all components require the same set of tasks. For example, this set of tasks could be design, tooling, fabrication and testing. Or to obtain finer detail, components could be categorized by type: heat exchanger, software, avionics, mechanical, sensors, actuators, etc. Each component category would have a different set of tasks. For example, a software "component" of a system would include tasks like design, code, and test while an avionics component may include define requirements, perform trade studies, perform analysis, design, build prototype, and test. Each component category, or bucket, will have a unique work transformation matrix. With this method, a set of tasks and work transformation matrices for each component category would have to be identified and each component would have to be associated with a category. This greatly reduces the time required to set up a task based model. Additionally, there can be a distinction in the set of tasks between a parent component and its subcomponents.

[11] This is defined by a convergence criterion. All elements of u_i must meet the convergence criterion in order to state that the u_i vector has converged to the zero vector.

For example, a parent component may need to be conceptually designed and later its subcomponents would have to be assembled and then tested. A typical work transformation matrix or process matrix produced in this way can be seen in Figure 18.

The total work vector, U, must converge to a finite value. A non-converging U suggests that the given the set of interdependent tasks cannot be completed in a finite amount of time. We know from experience that in nearly all cases, designs are completed in a finite amount of time. Convergence is really determined by the values in the work transformation matrix, WT. Because these values are estimates of some true value, it is possible to create a non-converging matrix. Therefore, the values in the matrix should be carefully chosen in order to create convergence. One way to create convergence is to incorporate the "learning effect." After each iteration, the task performer becomes slightly more efficient at their task. This can be represented by multiplying the entire WT matrix by a value less than 1 after each iteration. This value would need to be experimentally verified, but could be as high as 15% decrease in time required to finish the task.

Process Matrix

		CU0			CU1					CU2					CU3					CU4				
		Conceptually Design	Assemble	Test	Design	Tooling	Fabrication	Test	Installation	Design	Tooling	Fabrication	Test	Installation	Design	Tooling	Fabrication	Test	Installation	Design	Tooling	Fabrication	Test	Installation
CU0	Conceptually Design	0.00	0.00	0.10	0.05	0.00	0.00	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.05	0.00	0.00	0.00	0.00
	Assemble	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.00	0.00	0.00	0.00	0.10	0.00	0.00	0.00	0.00	0.10	0.00	0.00	0.00	0.00	0.10
	Test	0.00	0.00	0.00	0.10	0.00	0.00	0.00	0.00	0.10	0.00	0.00	0.00	0.00	0.10	0.00	0.00	0.00	0.00	0.10	0.00	0.00	0.00	0.00
CU1	Design	0.05	0.05	0.10	0.00	0.10	0.10	0.50	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Tooling	0.00	0.00	0.00	0.50	0.00	0.20	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Fabrication	0.00	0.00	0.00	0.00	0.50	0.00	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Test	0.00	0.00	0.00	0.70	0.00	0.40	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Installation	0.00	0.00	0.00	0.00	0.00	0.80	0.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
CU2	Design	0.05	0.05	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.10	0.50	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Tooling	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.00	0.20	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Fabrication	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.00	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Test	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.70	0.00	0.40	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Installation	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.80	0.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
CU3	Design	0.05	0.05	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.10	0.50	0.10	0.00	0.00	0.00	0.00	0.00
	Tooling	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.00	0.20	0.10	0.00	0.00	0.00	0.00	0.00	0.00
	Fabrication	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.00	0.10	0.00	0.00	0.00	0.00	0.00	0.00
	Test	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.70	0.00	0.40	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Installation	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.80	0.80	0.00	0.00	0.00	0.00	0.00	0.00
CU4	Design	0.05	0.05	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.10	0.50	0.10
	Tooling	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.00	0.20	0.10	0.00
	Fabrication	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.00	0.10	0.00
	Test	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.70	0.00	0.40	0.00	0.00
	Installation	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.80	0.80	0.00

Figure 18: A Complete Work Transformation Matrix using the same WTs for each component with one parent component.

Once the total work vector U [unit time] has been calculated, the total labor cost can be calculated. First, an estimate of the time it would take to complete the task independently from any other tasks is necessary, also known as the time to complete the first iteration. This vector, W0 [time per unit time], is then multiplied by U to determine the total work load vector, WL

[time]. The WL vector is the total amount of time required to complete each task, given all the interdependencies. The i^{th} component WL is multiplied by the i^{th} component of C, which contains the cost per hour of each task, to yield the i^{th} component of the total cost vector, TC. This process is summarized in Equation 10.

$$TC_i = WL_i \, C_i = U_i \, W0_i \, C_i$$

Equation 10

Equation 10 demonstrates the calculation of the final labor cost using the work transformation matrix in Figure 18. The first column represents u_0, a column vector of 1's. The total work vector, U, is calculated in the second column using Equation 9 and converges in 50 iterations. The vector, W0, which contains the time to complete the first iteration for each task was estimated in the third column. The vector, WL, which contains the total time to complete each task is in the fourth column. The cost vector, C, which contains the cost per hour for each task is in the fifth column. The vector, TC, which contains the total cost for each task, is in the final column. The sum of each element of the TC vector yields the total cost of the system given the work transformation matrix, WT, and estimates of W0 and C. If this development labor cost estimate does not seem to be accurate, then the W0 vector must be inaccurate. The model can be calibrated by adjusting the values of W0 until TC matches with either historical data or other estimates. This process of calibration tries to match the amount of iteration performed historically with the amount that will be performed with this design.

Component	Task	**u0** (units of time)	**U** for Design A (units of time)	**W0** - Time to Complete First Iteration (hours)	**WL** - Total Number of Hours Per Task	**C** - Cost Per Hour of Task ($/hour)	**TC** - Labor Cost of Task ($)
CU0	Conceptually Design	1	3.582	400	1432.79	$150.00	$214,918.67
	Assemble	1	6.856	40	274.22	$50.00	$13,711.05
	Test	1	5.138	100	513.76	$75.00	$38,532.16
CU1	Design	1	10.357	150	1553.60	$150.00	$233,039.55
	Tooling	1	8.511	150	1276.66	$75.00	$95,749.62
	Fabrication	1	6.326	75	474.42	$75.00	$35,581.70
	Test	1	10.768	20	215.36	$75.00	$16,152.12
	Installation	1	14.658	10	146.58	$100.00	$14,657.93
CU2	Design	1	10.357	150	1553.60	$150.00	$233,039.55
	Tooling	1	8.511	150	1276.66	$75.00	$95,749.62
	Fabrication	1	6.326	75	474.42	$75.00	$35,581.70
	Test	1	10.768	20	215.36	$75.00	$16,152.12
	Installation	1	14.658	10	146.58	$100.00	$14,657.93
CU3	Design	1	10.357	150	1553.60	$150.00	$233,039.55
	Tooling	1	8.511	150	1276.66	$75.00	$95,749.62
	Fabrication	1	6.326	75	474.42	$75.00	$35,581.70
	Test	1	10.768	20	215.36	$75.00	$16,152.12
	Installation	1	14.658	10	146.58	$100.00	$14,657.93
CU4	Design	1	10.357	150	1553.60	$150.00	$233,039.55
	Tooling	1	8.511	150	1276.66	$75.00	$95,749.62
	Fabrication	1	6.326	75	474.42	$75.00	$35,581.70
	Test	1	10.768	20	215.36	$75.00	$16,152.12
	Installation	1	14.658	10	146.58	$100.00	$14,657.93
	Total	**23**	**218.06**	**2160**	**16887.3**	**n/a**	**$1,847,885.56**

Figure 19: Sample labor cost calculation, based on the work transformation matrix in Figure 12.

Estimate the Cost Impact of a Design Change

In order to estimate the change in cost of a system due to a functional change, the magnitude and the phase of the change must be known.

The phase of the system is important in identifying in what phase of the design the change is being implemented. The phase of the system answers the question of "How much of the full design has been finished?" This can be thought of as the amount of FR-DP decomposition completed. If a change is made early in the design phase, when say 10% of the design is completed, then the design change should be relatively cheap to make. A change made late in the design phase, when say 90% of the design is completed, will be have a far higher cost penalty. American car companies, for example, sometimes find it necessary to make a design change when the car is already in the consumers hands due to dangerous failures. These recalls cost American car companies millions of dollars more than they would have had to pay if they had

made the change while early in the design phase. The phase of the change lies in the calculation of the total work vector. The cumulative sum of all the elements of U are plotted versus the number of design iterations, N, in Figure 20.[12] The total amount of work for all tasks that must be done is 220 units of time. Therefore, the phase of the design can be determined by dividing the current amount of time completed by the total amount of time. 10% of the design would be completed if 22 units of time had been completed, 20% of the design would be completed if 44 units of time had been completed, and so on.[13]

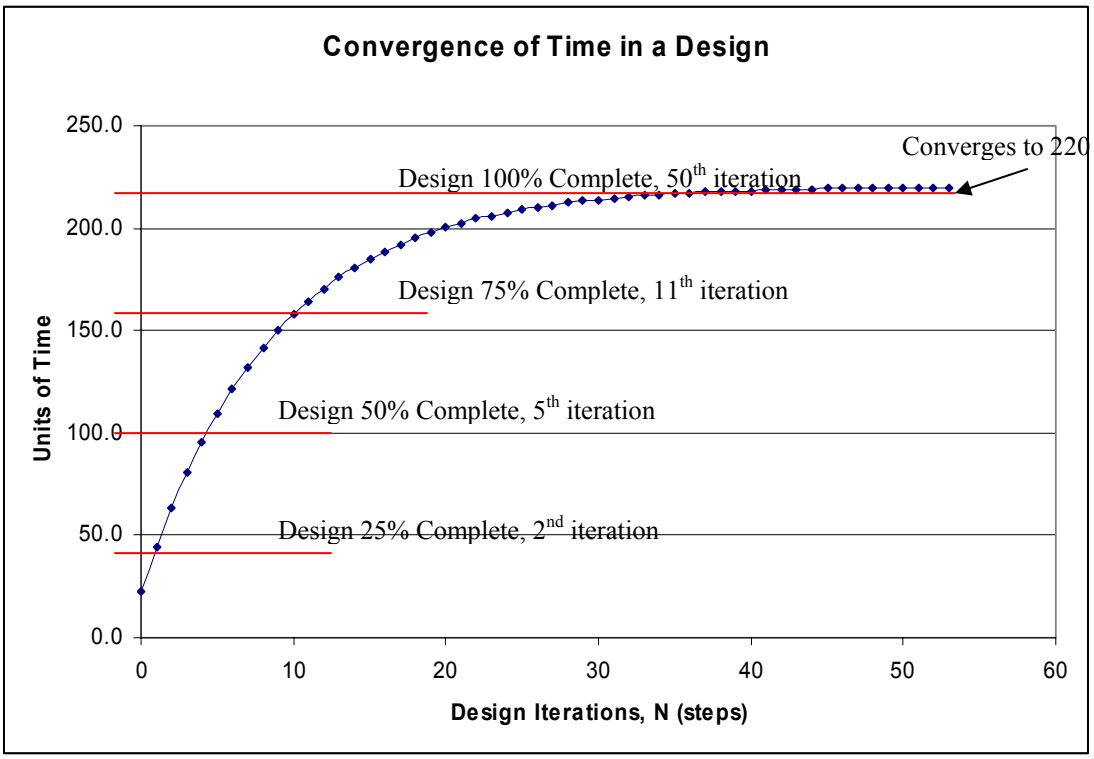

Figure 20: Seeing the phase of the design in the total work vector, U. Please note that these are units of time, before multiplication of W0, and not actual hours.

The magnitude of the change comes from the percentage rework that was calculated from Figure 15. This tells us how much more work must be done in order to complete the new design. Recall that the percentage rework is calculated for each of the CUs, not each task. Therefore, the percentage rework vector will look something like the following:

$$\%\ \text{Rework} = \boldsymbol{\alpha} = \{0.25, 0.25, 0.25, 0.25, 0.67, 0.67, 0.67, 0.67, \ldots, 0.33, 0.33, 0.33, 0.33\}^T$$

where the tasks corresponding to the same CU have the same % rework factor.

Mathematically speaking, the actual calculations work as follows. Rewriting Equation 10:

$$TC_i = U_i\ W0_i\ C_i$$

Equation 11

[12] This graph is obtained used the following method: for example step 0 is the sum of all elements in u_0, step 10 is the sum of all elements in u_0, u_1, …, and u_{10}. This is a measure of the total amount of work completed in units of time as the number of design iterations or steps increases.

[13] It should be noted that, in the end, each unit of time for each task is weighted differently by the W0 vector. The best way to determine the design completion would be to compare with the weighted times, and not the unweighted times. This is not done in this analysis.

If we make a change at the j^{th} iteration, with a rework vector, **α**, then the i^{th} component of u_j becomes

$$u^*_{j_i} = (1-\alpha_i)u_{j_i} + \alpha_i$$

Equation 12

This means that of the work that has been completed by the j^{th} iteration, **α**% of that work must be completely redone. This will affect the total work vector, U.

$$U^* = \left(I + WT + \ldots + WT^j\right)u_0 + \left(WT^{j+1} + \ldots + WT^N\right)u^*_j$$

Finally, substitute this new work vector, U^*, into Equation 11.

$$TC_i = U^*_i \; W0_i \; C_i$$

Equation 13

Suppose that for the same example from Figure 18, Figure 19 and Figure 20, the % rework for CU0, CU1, CU2, CU3, and CU4 was determined to be 25%, 50%, 75%, 100%, 25%, respectively. Also assume that we decided to make a change from design A to design B when about 60% of the design had been completed by the 7^{th} iteration. The time to complete the system design is plotted in Figure 21, as done before in Figure 20. Notice that making this change requires an additional 70 units of time to complete. The final results are calculated in Figure 22.

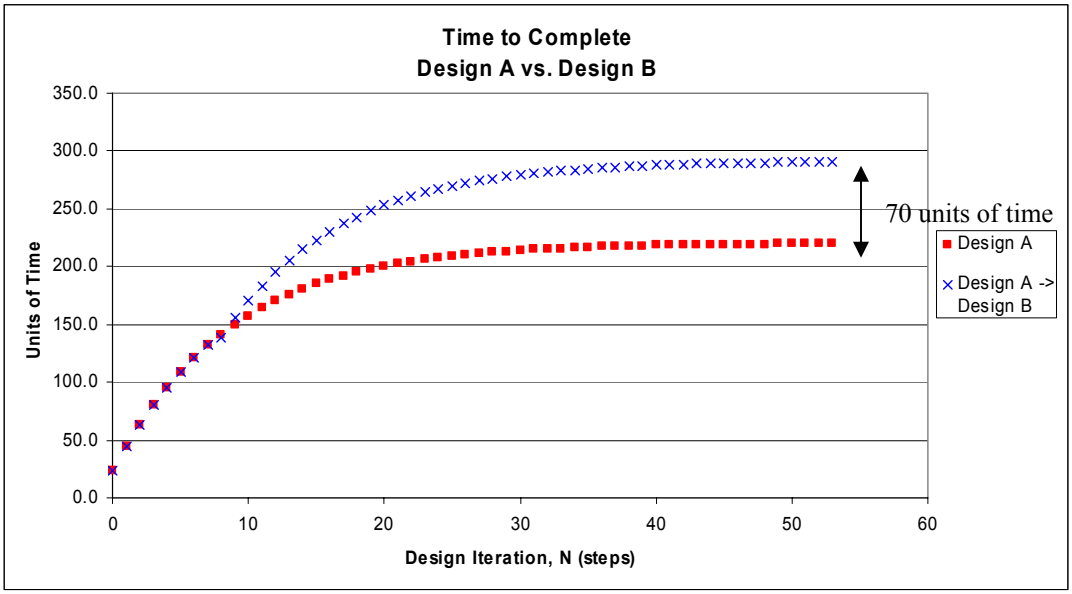

Figure 21: The impact on the total amount of time to complete the design when making a change from design A to design B at 60% completion

Component	Task	**u0** (units of time)	**u7** (units of time)	**α** (% rework)	**u7*** (units of time)	**U*** for Design A changed to Design B (units of time)	**W0** - Time to Complete First Iteration (hours)	**WL** - Total Number of Hours Per Task	**C** - Cost Per Hour of Task ($/hour)	**TC** - Labor Cost of Task ($)
CU0	Conceptually Design	1	0.142	0.25	0.356	4.638	400	1855.14	$150.00	$278,270.47
	Assemble	1	0.334	0.25	0.500	8.963	40	358.51	$50.00	$17,925.46
	Test	1	0.232	0.25	0.424	6.699	100	669.92	$75.00	$50,244.30
CU1	Design	1	0.515	0.50	0.758	13.279	150	1991.79	$150.00	$298,768.85
	Tooling	1	0.416	0.50	0.708	10.950	150	1642.45	$75.00	$123,183.45
	Fabrication	1	0.292	0.50	0.646	8.207	75	615.49	$75.00	$46,161.73
	Test	1	0.537	0.50	0.769	13.804	20	276.09	$75.00	$20,706.49
	Installation	1	0.743	0.50	0.872	18.731	10	187.31	$100.00	$18,730.61
CU2	Design	1	0.515	0.75	0.879	14.268	150	2140.15	$150.00	$321,022.34
	Tooling	1	0.416	0.75	0.854	11.848	150	1777.14	$75.00	$133,285.44
	Fabrication	1	0.292	0.75	0.823	8.943	75	670.69	$75.00	$50,301.95
	Test	1	0.537	0.75	0.884	14.907	20	298.13	$75.00	$22,359.91
	Installation	1	0.743	0.75	0.936	20.265	10	202.65	$100.00	$20,265.27
CU3	Design	1	0.515	1.00	1.000	15.257	150	2288.51	$150.00	$343,275.82
	Tooling	1	0.416	1.00	1.000	12.746	150	1911.83	$75.00	$143,387.44
	Fabrication	1	0.292	1.00	1.000	9.679	75	725.90	$75.00	$54,442.18
	Test	1	0.537	1.00	1.000	16.009	20	320.18	$75.00	$24,013.34
	Installation	1	0.743	1.00	1.000	21.800	10	218.00	$100.00	$21,799.92
CU4	Design	1	0.515	0.25	0.636	12.290	150	1843.44	$150.00	$276,515.37
	Tooling	1	0.416	0.25	0.562	10.052	150	1507.75	$75.00	$113,081.45
	Fabrication	1	0.292	0.25	0.469	7.470	75	560.29	$75.00	$42,021.50
	Test	1	0.537	0.25	0.653	12.702	20	254.04	$75.00	$19,053.06
	Installation	1	0.743	0.25	0.807	17.196	10	171.96	$100.00	$17,195.95
	Total	**23**				**290.7**	**2160**	**22487.3**	**n/a**	**$2,456,012.31**

Figure 22: Labor cost calculation given a change from design A to design B at 60% design completion

The cost of implementing design B at 60% completion of design A (at the 7th iteration step) with the rework vector as previously mentioned will cost an additional $608,126.75 for the development phase in labor.[14]

From historical data, it is known that the labor cost is a certain percentage of the total development cost. This percentage varies depending on the type of component. For example, for a software application, labor required for development may be 90% of the total development costs. For a heat exchanger, perhaps the labor required for development may be 50% of the total

[14] This is calculated by subtracting the total cost of design A from the total cost of design B.

development cost. This rule of thumb gives an easy calculation of the total cost of a component. The way of calculating the total labor cost of a component in this cost estimation method is to sum the total cost of all the tasks required to complete that component. Then the total development cost of the component can be calculated (recall from Equation 2 that this includes both labor and material costs):

$$\text{Total Cost of Component} = \frac{\text{Total Labor Cost of Component}}{\text{\% Labor Cost of Total Cost}}$$

Equation 14

For the example that we have been following, suppose that development labor cost is typically 50% of the total development cost. The total development cost is then determined by using Equation 14. Without the design change, the total development cost is $3,695,771.12. With the design change, the total development cost is $4,912,024.62. The suggested design change, therefore, has an additional total development cost of $1,216,253.50.

Summary

The method outlined in this chapter provides a quick way of estimating the cost impact of a design change. The procedure is as follows:

1. Determine the list of components affected by a change
    a. Determine the list of FRs that will change due to changes in constraints or customer needs.
    b. Determine the list of DPs that are necessary to change in order to satisfy the changed FRs by using the design matrix.
    c. Determine the list of components that are affected by using the DP-CU Matrix.
    d. Further refine the list of components that are affected by examining the physical relationship between the changed components and other components, using a CU-CU matrix.
2. Determine the amount of time required to make the changes to the list of affected components
    a. Determine the total cost without the design change by using Equation 10.
    b. Determine the % Rework for each CU.
    c. Specify the phase of the design change.
    d. Determine the total cost with the design change by using Equation 13.
3. Compare and analyze the total development cost before and after the change.

The ability to quickly estimate the cost impact of design changes is invaluable as a medium of communication between customers and developers. Recall the case of Ingalls and the Navy presented in Chapter 1. The customer, the Navy, repeatedly asked for changes to the design without realizing the resonating effect it was creating throughout the entire development process. By the time they realized the effect of the design changes, Ingalls had exceeded budget by $2.7 billion. With a tool that can quickly show the impact of design changes, situations like this can be either avoided or anticipated and then negotiated. It will also aid in the decision making progress because it will create the opportunity to quickly be able to compare the new life-cycle cost to the benefits of making that change. Decision-makers can make decisions with more information and confidence about the advantages of making a design change.

The largest amount of time will be spent developing the model with the FR-DP, DP-CU, CU-CU, and Task Matrices. But this information, once gathered, has many applications beyond cost estimation. It can become an incredible resource to understanding not only the life-cycle cost, but also the design and organizational behavior. Axiomatic design is in itself a very effective

design tool that is a great improvement over current methods of design. Using axiomatic design will surely improve the overall design of the system. Also, there is a huge demand for task-based models in many organizations. It can be used to identify problems in the organization of projects and the design process. Please see the references provided on axiomatic design and design structure matrices for further applications.

If developers already use axiomatic design in their design process, this methodology will be an easy extension of the information that they have already generated. This practice is highly recommended. Attempting to record the designing process using axiomatic design, is in effect just designing and then trying to determine what you designed through the Axiomatic design perspective. This is, in effect, designing the system twice. Axiomatic design is extremely effective as a design methodology and was not developed with the intention of recording designs.

Chapter 4

Operations Cost

What makes a car reliable? Often people have a good sense of how reliable the car they drive is. A car owner pays an expected amount to maintain the car, as prescribed by the maintenance, and the car performs as expected. If the car does not perform as expected, or the owner of the car has to pay more than expected, then the car is thought to be unreliable. In daily life, we have a good sense of the cost of operations. How do the designers of the car estimate the cost of operation for a system as large as a car? It is important to be able to estimate because it is critical to customers. People will often pay more for higher reliability, rather than pay for the high cost of operation.

An Introduction to Complexity

The key element in the cost of operation is time-dependent complexity. Recall from chapter 2 that complexity is the uncertainty in the design parameters (DPs) ability of satisfying the functional requirements (FRs). In axiomatic design, complexity takes on four different forms:

- Time-independent real complexity
- Time-independent imaginary complexity
- Time-dependent combinatorial complexity
- Time-dependent periodic complexity

Time-independent real complexity is the uncertainty that an FR will fall within the design range. It can be measured for each FR_i as the area beneath the probability density function (p.d.f.) of FRi within the design range. It is often measured by the information content. The range for which the p.d.f. of FR_i and the design range overlap is called the common range. Time-independent real complexity means that the common range does not change over time. See Figure 23. According to the Information Axiom, a good design is one that has the least information content (uncertainty). Therefore, the designer should always be diligent in keeping all FRs within the design range.

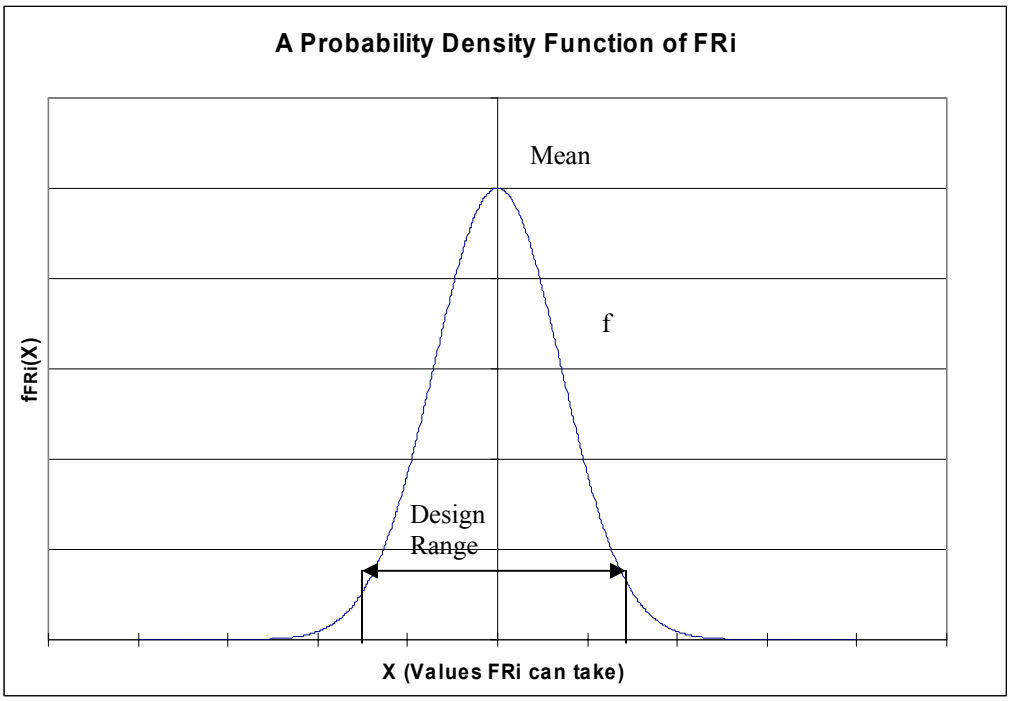

Figure 23: A graph of an FR with time-independent real complexity. The p.d.f. of FR_i and the design range do not change with time.

Time-independent imaginary complexity is a result of the lack of knowledge of a design. This uncertainty is difficult to measure, since it is not possible to measure what is not known. Suppose a designer developed a design that had the following design matrix:

$$\begin{Bmatrix} FR_1 \\ FR_2 \\ FR_3 \\ FR_4 \end{Bmatrix} = \begin{bmatrix} X & 0 & 0 & x \\ X & X & 0 & 0 \\ 0 & X & X & 0 \\ X & 0 & 0 & X \end{bmatrix} \begin{Bmatrix} DP_1 \\ DP_2 \\ DP_3 \\ DP_4 \end{Bmatrix}$$

Equation 15

Let the large X's denote the known FR-DP dependencies and the small *x* denote the unknown dependency. The designer would think that all that was necessary to complete the design was to modify the DPs in the proper sequence: DP_1, DP_2, DP_3, then DP_4. However, the designer does not know that DP_4 affects FR_1. By the time the sequence had been completed, FR_1 would no longer be satisfied. This uncertainty in being able to fulfill all FRs is related to the lack of knowledge of the design. The only way to reduce imaginary complexity is through research or experience. Only then can the unknown values of the A_{ij} components of the design matrix be determined.

Time-dependent combinatorial complexity means that the uncertainty in a design increases in time. This is typically experienced during the operation of a designed product. The designed product during operation experiences different effects such as fatigue, corrosion, friction, impacts, fouling, etc. Or in the case of software, these effects may include memory leaks, data corruption, etc. The overall result of these degrading effects is that the FRs fall farther and farther out of the design range. The degrading effects combine and, therefore, the uncertainty increases in time. Suppose that an FR has a normal p.d.f. with mean μ_0 and standard deviation σ_0. As time progresses, a degrading mechanism can increase the mean, as seen in Figure 24, or the standard

deviation, as seen in Figure 25. As a result, the common range decreases in time. Actually, in real-life applications, the change of the PDF over time is a combination of changes to the standard deviation and to the mean. The change in time of uncertainty due to degrading mechanisms is combinatorial complexity.

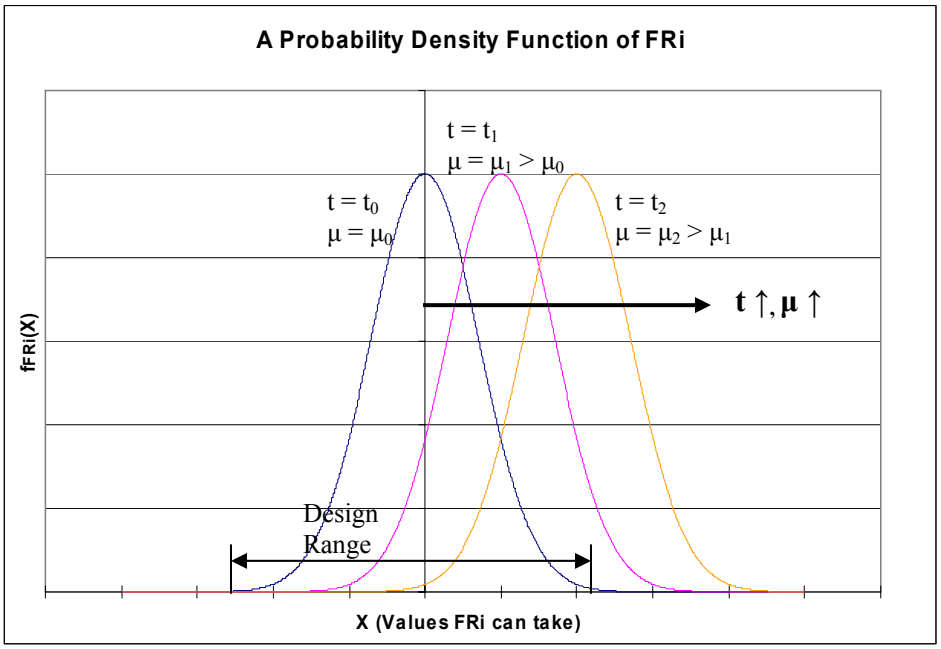

Figure 24: The drift of the mean increases uncertainty over time.

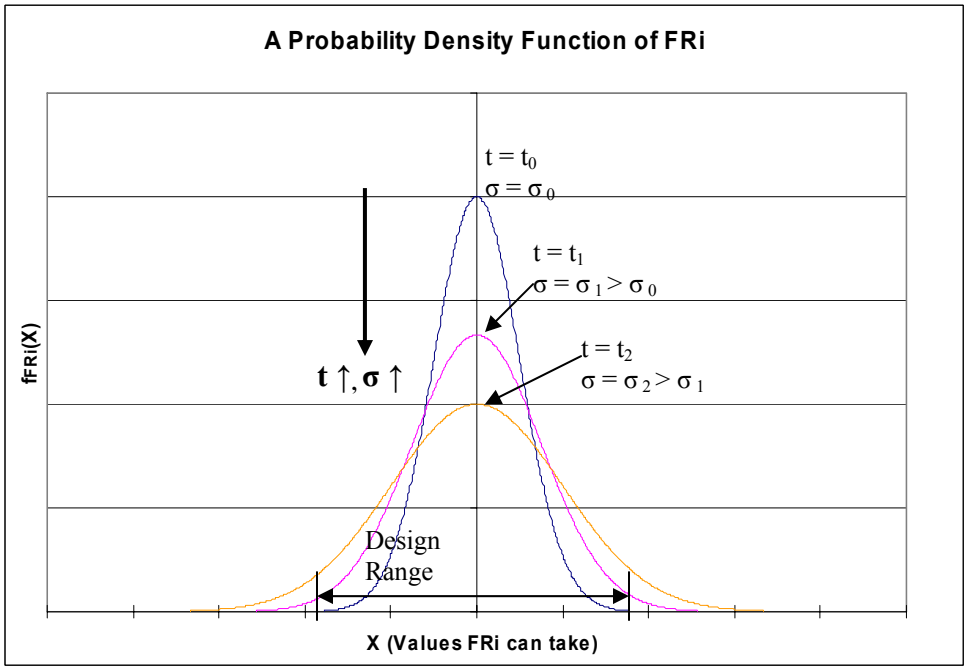

Figure 25: The drift of the standard deviation increases uncertainty over time.

If this is what most FRs experience during normal operation of the system, then how is it that designed products do not break apart completely in a short amount of time? The reason is because of time-dependent periodic complexity. Periodic complexity is basically combinatorial complexity with reinitialization. Car owners know periodic complexity as the periodic

maintenance schedule in the owner's manual, but periodic complexity is far more widespread than that. Periodic complexity is necessary in all living creatures. The cell cycle is one form of periodic complexity. A single cell splits into two cells in a very complicated cycle, which causes cells to be reinitialized periodically. A human life is another example in which a human experiences various outside degrading mechanisms: exposure to harmful radiation, toxins, bacteria, viruses, accidents. A natural death is one in which combinatorial complexity causes a vital organ to malfunction. Without reproduction, a form of periodic complexity, the human race would be finished. But instead every 20-30 years a new generation of humans are produced, and humankind perseveres. In another example, suppose that an operating system had been running for several days. In that time span, millions of operations had been carried out, but several programs had crashed. Due to combinatorial complexity, eventually the operating system crashes. However, if we had to buy a new computer after that, the computer industry would not be thriving as it is today. Instead, we hit the reboot button and the entire operating system is reinitialized, the memory is wiped clean, and all programs can assume normal operation. The key element in periodic complexity, therefore, is reinitialization.[15]

Axiomatic Design of Operations

The operation of a designed system is subject to time-varying complexity and must be reinitialized in order for all FRs to remain fulfilled. Operation should be accounted for during the design with the following procedures:

1. Identify all DPs that cause the combinatorial complexity of FRs.
2. Attempt to eliminate combinatorial complexity by transforming it into time-independent real complexity with zero information content or time-dependent periodic complexity.
3. Each DP that is still subject to combinatorial complexity implies a new operational FR that states: "FR_i should be maintained within the design range (FR_{min}, FR_{max})." The DP to solve this FR should be "Maintain DP_1, …, DP_N within a specified range." This introduces periodicity.
4. An operation should be designed according to this newly created FR.

How do we identify the FRs undergoing combinatorial complexity processes and the DPs that are causing it? We can determine this by watching the mean and variance of an FR over time. Assume that the design is linear. We know that any FR is given by Equation 16.

$$FR_i = \sum_j A_{ij} DP_j$$

Equation 16

The expected value of an FR is, therefore, given by Equation 17.

$$E[FR_i] = \mu = \sum_j E[A_{ij} DP_j]$$

Equation 17

If we take the derivative in time of Equation 17, as seen in Equation 18, we can determine which FRs will change their mean over time, and by which DPs. The DPs that are independent of time will drop out of equation, leaving only the DPs that change in time. It is also interesting to note that even if DP_j is independent of time, but A_{ij} is dependent on time, then it can still change the mean over time.

[15] See references [7] and [14] for extensive discussions on periodic complexity and complexity in general.

$$\frac{d\mathrm{E}[\mathrm{FR_i}]}{dt} = \frac{d\mu_i}{dt} = \sum_{\mathrm{j}} \frac{d\mathrm{E}[\mathrm{A_{ij}DP_j}]}{dt}$$

Equation 18

However, even if the mean does not change with time, the variance still may change with time. Therefore, it is also necessary to analyze how the standard deviation of FR_i changes with time. Let us assume for simplification that all DP_js are independent of each other and that A_{ij}s are constant. The variance of FR_i can be seen in Equation 19. The standard deviation of a general FR_i, σ_i, is given by Equation 20.

$$\mathrm{var}(\mathrm{FR_i}) = \mathrm{var}\left(\sum_{\mathrm{j}} \mathrm{A_{ij}DP_j}\right) = \sum_{\mathrm{j}} \mathrm{A_{ij}}^2\ \mathrm{var}(\mathrm{DP_j}) = \sum_{\mathrm{j}} \mathrm{A_{ij}}^2 \left(\mathrm{E}[\mathrm{DP_j}^2] - (\mathrm{E}[\mathrm{DP_j}])^2\right)$$

Equation 19

$$\sigma_i = \sqrt{\mathrm{var}(\mathrm{FR_i})} = \sqrt{\sum_{\mathrm{j}} \mathrm{A_{ij}}^2 \left(\mathrm{E}[\mathrm{DP_j}^2] - (\mathrm{E}[\mathrm{DP_j}])^2\right)}$$

Equation 20

If we take the derivative in time of Equation 20, as seen in Equation 21, we can determine which FRs will change their standard deviation over time, and by which DPs. By similar argument as above, time independent DPs will drop out of the equation and leave the time dependent DPs.

$$\frac{d\sigma_i}{dt} = \frac{\frac{d\mathrm{E}[\mathrm{DP_j}^2]}{dt} - 2\mathrm{E}[\mathrm{DP_j}]\frac{d\mathrm{E}[\mathrm{DP_j}]}{dt}}{2\sqrt{\sum_{\mathrm{j}} \mathrm{A_{ij}}^2 \left(\mathrm{E}[\mathrm{DP_j}^2] - (\mathrm{E}[\mathrm{DP_j}])^2\right)}}$$

Equation 21

Essentially, if $\frac{d\mu_i}{dt} \neq 0$ or $\frac{d\sigma_i}{dt} \neq 0$, then FR_i has combinatorial complexity. If this is the case, then either FR_i must be redesigned to such that $\frac{d\mu_i}{dt} = 0$ and $\frac{d\sigma_i}{dt} = 0$ or an operation must be developed to maximize the common range of FR_i. Thus, combinatorial complexity of FR_i implies a new $FRop_i$ that states "Maintain FR_i within the design range ($FR_{i(min)}$, $FR_{i(max)}$) (or maximize the common range)." The DPopi is also implied: "Stabilize the DPs that contribute to the combinatorial complexity of FR_i." Then by zigzagging, one can create sub-FRs that address each of the DPs that contribute to the combinatorial complexity separately. By performing periodic reinitialization, the combinatorial complexity is guaranteed to be transformed into time-dependent periodic complexity and long-term stability can be achieved.

Cost of Operation

So now that it is established that each DP whose variance and mean changes with time necessarily must have an operation to stabilize it during use, we can address the question of, what is the cost of operation? The key lies in the frequency of the reinitialization, the cost of reinitialization, and the cost of maintaining the capability of reinitialization per unit time. How often do we reinitialize the system, how much does it cost us per reinitialization, and how much

does it cost us while we are not reinitializing the system? One could also argue that there is an initial investment required to setup the capability of reinitialization, but this also can be captured in the development cost. Define \$(g) as the cost of g, where g is an function. The total cost of an operation_i depends on time (how long the function is used), seen in Equation 22, where $f_{\text{reinitialization}}$ is the frequency of reinitialization. The total life-cycle cost of all operations can be calculated by integrating the cost of operations from t_0 to $t_{\text{life-cycle}}$, as seen in Equation 23. Note that the cost of reinitialization is an all encompassing term that includes the cost of consumables. For example, the functional requirement of "Store fuel to be supplied to the engine" is fulfilled by a fuel tank filled with gasoline. Once this fuel tank becomes empty, it can no longer fulfill its original FR and must be reinitialized by refueling.

$$\text{\$(operation}_i\text{(t))} = (\ \text{\$(reinitialization)}\cdot f_{\text{reinitialization}} + \text{\$(maintaining capability)}\ )\cdot t$$

Equation 22

$$\text{\$(life-cycle operation)} = \int_{t_0}^{t_{life-cycle}} \$(operation_i(t))dt$$

Equation 23

The cost and frequency of reinitialization are dependent on the design. For example, suppose that a part from an engine bay had to be reinitialized by replacing the part. The operation cost is a function of how the part was placed inside the engine bay, how it was mounted, how large it is, etc. A simple engine bay layout would imply a simple operation of replacing the part, whereas a very complicated engine bay layout would imply a complex operation of replacing the part. The frequency of reinitialization is also dependent on the design. How often the part is replaced depends on the mechanisms of wear: fouling, friction, fatigue, etc. Extensive testing is often needed in order to determine an approximate frequency of reinitialization, since it is often difficult to calculate or simulate.

Designers can specify a reasonable frequency of reinitialization, but the user has the ultimate opinion on the matter. There is a certain sense of subjectivity in the frequency of reinitialization. The reason for this is because different people have a different value of functions. Let us take a keyboard for an example. Suppose the z-key had begun to malfunction. Because this letter is the least used letter in the English language, the user may decide to wait on fixing the keyboard. The cost of reinitializing the z-key would be either to call a technician to fix the broken circuit or software or simply to buy a new keyboard. Reinitialization can wait, because the cost of reinitializing is far greater than the cost of a new keyboard. However, if the a-key had begun to malfunction, then the user would consider buying a new keyboard since it would cripple the user's ability to communicate. The cost of reinitializing the a-key is the same as reinitializing the z-key. Now suppose that the user was using the keyboard to type in PRC Pinyin to type simplified Chinese characters. The frequency of the letter z in the Chinese language is far more frequent than in English. Therefore, when the z-key breaks, a Chinese user may decide to replace the keyboard. The value of functions to each user is subjective.

What is the frequency of maximum efficiency (or minimum cost)? It is altogether related to the probability of failure. The probability of failure of any particular FR is given by Equation 24, where DR_{min} and DR_{max} are the bounds of the design range. Since $f_{\text{FR}i}$ is a function of time, then $\text{P}_{\text{failure}}$ is also a function of time. You can see from Equation 24 that as the common range decreases, the probability of failure increases.

$$\text{P}_{\text{failure}}(t) = 1 - \int_{\text{DR}_{\text{min}}}^{\text{DR}_{\text{max}}} f_{\text{FR}_i}(x,t)dx$$

Equation 24

The optimal reinitialization frequency can be found by solving Equation 25 for $f_{optimal}$. This is when the cost of reinitialization is equal to the cost of the function failure multiplied by the probability of failure.

$$P_{failure}\left(t_{optimal}\right) = P_{failure}\left(\frac{1}{f_{optimal}}\right) = \frac{\$(\text{reinitialization})}{\$(\text{function failure})}$$

Equation 25

In order for us to examine why this is so, let us examine a simple case. Suppose that the probability of failure is 0 before t0, then increases linearly with time, reaching a final value of 1 at t_1. Then the cost of a functional failure is just a scaling of the probability of failure. The value of $(functional failure) does not change with time, unless the user changes the value of the function over time. The cost of reinitialization is a cost defined by the designer, and is thus constant to the user. Suppose the cost of functional failure as a function of time curve for one user is seen in Figure 26. If the cost of reinitialization were $(reinitialization)$_1$, then the optimal frequency of reinitialization would be $\frac{1}{t_{optimal}}$, where $t_{optimal}$ is at the point of intersection of $(reinitialization)$_1$ and P(t)$(function failure). Another interesting case could be when the cost of reinitialization never intersects P(t)$(function failure). In this case, it would always be more expensive to reinitialize than have the function fail.

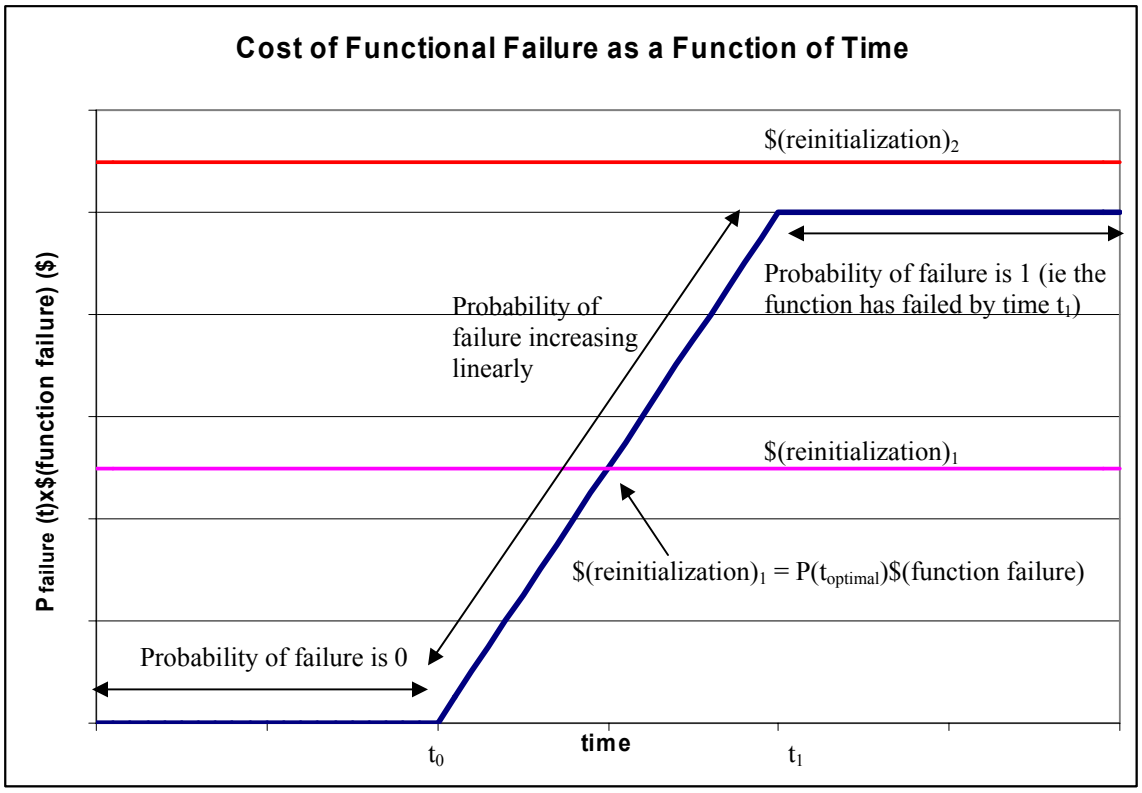

Figure 26: The cost of a functional failure compared to the cost of reinitialization.

For the case of the a-key and z-key, assume that the $(reinitialization) and P(t) (not exactly true) are the same for the two, and that $(a-key failure) > $(z-key failure). We could see then that the user from Figure 27 that the user would change the keyboard at time $t_{optimal}$ if the a-

key broke and would never change the keyboard if the z-key broke. The designer of the keyboard probably would suggest the change of the keyboard at time t_0. However, this is not the optimal time of reinitialization.

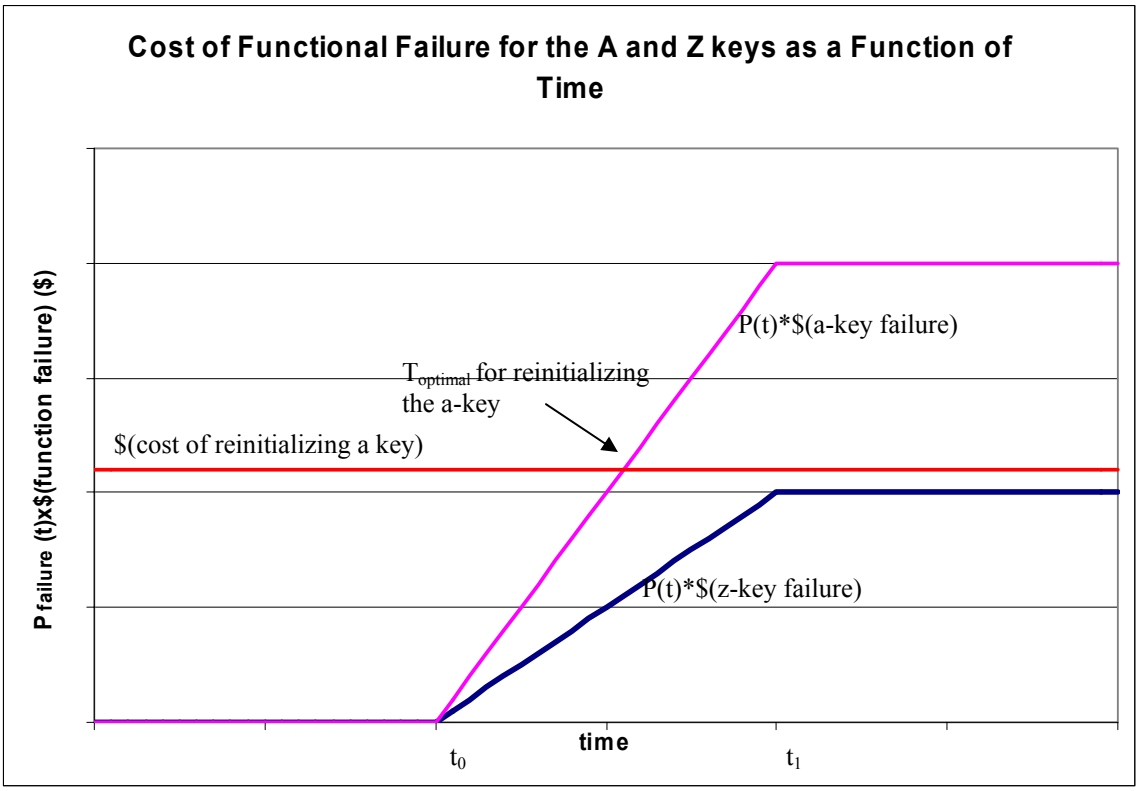

Figure 27: The effect of $(a-key failure) > $(z-key failure) .

This analysis is handy, but often it is difficult to directly calculate $(function failure) due to the matter of subjectivity. It all depends on the value of functions, and the amount of risk that the user can tolerate. Ultimately, the designer should probably suggest that the frequency of reinitialization be t_0, and then allow the user to decide how much longer after that to actually perform the reinitialization.

Operations are defined by the reinitialization procedures that are necessary for FRs that have combinatorial complexity. The cost of these operations is completely characterized by the frequency of reinitialization, cost of reinitialization, and the cost of maintaining the capability of reinitialization. This fits into the goal of determining the cost impact of a design change quite nicely. Once the design has been decomposed to a significant level with operations embedded into the design, it is a matter of determining which DPs are affected by which FRs. As before, the input is a list of FRs that will change, and the output is a list of DPs that are affected. These DPs could include operational DPs. Therefore, it is easy to see which operations are affected by an FR change. The next task is to determine the change in cost to the operations. Before the design change, suppose that all the operational cost parameters, $f_{\text{reinitialization}}$, $(reinitialization), and $(maintain capability) are known. Then it remains to determine what the changes in each of these parameters are due to the design change. Let $(operation) be $(op), $f_{\text{reinitialization}}$ be f_r, $(reinitialization) be $(r), and $(maintaining capability) be $(m). The change in cost would then be calculated by Equation 26.

$$\$(op_2) - \$(op_1) = \Delta\$(op_{2\text{-}1}) = [\$(r_2)\cdot f_{r2} - \$(r_1)\cdot f_{r1} + \$(m_2) - \$(m_1)]\cdot t$$

$$\Delta\$(op_{2\text{-}1}) = [\$(r_2)\cdot f_{r2} - \$(r_1)\cdot f_{r1} + \Delta\$(m_{2\text{-}1})]\cdot t$$

Equation 26

Summary

It is important to be able to estimate the cost of operation of a system during the design phase. Customers will pay a higher upfront cost for a lower operation cost over the long run. For example, NASA will pay top dollar to re-design certain aspects of the space shuttle in order to lower the cost of operation.

Vital to understanding the cost of an operation is the knowledge of how the design of the system affects the design of operation. Operations arise due to the fact that FRs undergo combinatorial complexity. Operations effectively introduce periodicity into a system with combinatory complexity. FRs undergo combinatorial complexity because the mean or standard deviation of the DPs that affect that FR change over time due to degrading mechanisms. FRs that have combinatorial complexity must either be redesigned to eliminate time-dependent complexity, or an operation must be designed in order to introduce periodic complexity. If FR_i has combinatorial complexity, then it implies an $FRop_i$ that states "Maintain FR_i within the design range (DR_{min}, DR_{max}) (maximize the common range)." The corresponding $DPop_i$, then, is "Stabilize the DPs that contribute to the combinatorial complexity of FR_i." The operation can be designed by decomposing the $DPop_i$ further to address each DP separately. The purpose of each operation is clearly stated. Any operation that does not maximize the common range of a functional requirement is superfluous.

The cost of an operation is defined by three parameters: the frequency of reinitialization, the cost of reinitialization, and the cost of maintaining the capability of reinitialization. The frequency of reinitialization is dependent on the probability of failure of a function and the value of the function. The optimal frequency of reinitialization is when the probability of functional failure multiplied by the value of the function is equal to the cost of reinitialization. The impact of a design change can be determined by determining what the change to each of these three parameters is due to the specified design change. For the case of development cost, a method of determining the change in development cost was presented in Chapter 3. A method for determining the change of cost to the operational parameters has not yet been determined, but is a focus of future research.

Chapter 5

Key Cost Drivers

What are the key cost drivers of a system? Being able to identify key cost drivers is essential in cost minimization efforts. The best example comes from the first generation Space Shuttle. NASA's Budget 2002 states that seven flights were anticipated to cost $2,530,900,000, or $360,000,000 per flight, in operations cost.[16] What are the key cost drivers of the operations cost? Key cost drivers and improving safety are the main reasons that NASA in 2002 planned to spend $468,000,000 in shuttle upgrades. By spending $468,000,000 in 2002, NASA hoped to save operation costs and lives over the next decade that it had anticipated the orbiter to remain in use. NASA has had nearly 25 years of experience operating the space shuttle, with over 100 flights. How could they have known what the key cost drivers would have been 30 years ago when they were designing the orbiter? If they had this information, then they would have designed the system correctly 30 years ago, and would no longer need to upgrade the system. Therefore, 30 years ago, this information would have been worth $468,000,000 (2002) in cost savings.

There are two types of key cost drivers that so far have been identified: 1) the most expensive functional requirements (FRs), and 2) the amount of design iteration. The most expensive FRs offer a view to which functions of the system are the biggest source of expense. This cost-model achieves this by mapping the cost of each costing unit back into the FR-DP domain in a method called rolling up. The key cost driver of the second type indicates that the design is coupled. Two strategies for minimizing the impact of this key cost driver are to either remove the coupling or lessen the time it takes to iterate.

The Cost Of Functional Requirements

By knowing the cost of all FRs, we can easily see which functions are most costly. The roll-up method basically takes cost information from the costing unit domain and maps into each DP. Recall the DP-CU matrix described in Chapter 3. The DP-CU matrix came in handy when trying to identify which components will have to change as a result of a functional requirement change. It can also be used to transfer information from the costing unit domain into the functional domain. Often companies have cost estimates that exist in the costing unit domain and are based on a work breakdown structure. This cost information can be easily utilized with a DP-CU matrix to cost out the FR-DP map.

Recall the DP-CU Matrix from Chapter 3, seen in Figure 28. In chapter 3, the elements of the DP-CU matrix were X's. For the purpose of the roll-up method, the element A_{ij} of the DP-CU matrix denotes the percentage of the cost of CU_j that contributes to the cost of DP_i. For example, in Figure 28, $\$(DP_1) = 0.4\$(CU_1)$ and $\$(DP_2) = 0.6\$(CU_1) + 0.2\$(CU_2)$ and so on. The values of all the A_{ij}s must be determined by someone who understands both the design and the cost of each component. In a sense, this is a forced and subjective method of costing the FR-DP map, but it is the only method available at this point.

[16] See p. 19 of reference [15].

Development	CU_1	CU_2	CU_3
DP_1	0.4	0	0
DP_2	0.6	0.2	0
:	:	:	:
DP_6	0	0	1
:	:	:	:
DP_n	0	0.8	0

Figure 28: DP-CU used to "roll-up" costing information in the costing unit domain to the FR-DP domain.

Once the cost of DPs are established, then the task remains to cost out each FR. Recall that one DP can affect several FRs. One DP could only slightly affect one FR, but heavily affect another FR. Our intuition would say then, that the cost of that DP must transfer mostly to the heavily affected FR. Therefore, a method of determining the relative magnitude of a DPs affect on the FRs is needed in order to apportion the cost of the DP properly among the FRs. This will be the task of future research.

With the cost of FRs determined, one could analyze the FR-DP hierarchical map to see which functions of the design are most costly. This could be useful, for example, if one were deciding whether to make a design change or not. A design change that required the change of more expensive functions will most likely be a very costly change. It would also serve as an aid in understanding why the components that are affected by these FRs are costing so much.

Design Iteration

Design iteration is also another type of key cost driver that heavily impacts the development cost. There are two methodologies that are helpful when thinking about design iteration: 1) axiomatic design and 2) design structure matrices. Axiomatic design requires the designer to “design it right the first time” by removing all the coupling that is causing design iteration. The design structure matrix process accepts the fact that there is coupling, but tries to most effectively deal with that coupling. Both schools of thought will be discussed here.

Axiomatic Design - Discussion on Design Iteration

Design iteration, as defined in an Axiomatic design context, is caused by small design ranges, coupling, and imaginary complexity.

In the design process, one tries to maximize the overlap of the design range and system range (thereby minimizing the real complexity) by varying the DPs. Most engineers know that maintaining large tolerances in their DPs will require less expensive processes. Small tolerances will require expensive precision manufacturing processes, with the probability of failure being higher. The same is true for the design range of a functional requirement. If the design range is too small, then it will take many tries to adjust the DPs so that the system range falls within the design range. Keeping all the design ranges relatively large will lessen the amount of design iteration.

Coupling in a design also causes iteration. If a design matrix is coupled, then the designer will have to vary the DPs iteratively in order to maximize the overlap of the system ranges of all the FRs within their respective design ranges. Dan Frey and Nicolas Hirschi performed a study that tested the ability of people to solve coupled designs verses uncoupled designs.[17] They found that coupled designs took significantly more time to solve than uncoupled designs did, depending

[17] See reference [18]

on the number of variables involved. Their experimental results are summarized by Figure 29. As the number of variables increases, the amount of time to complete a coupled design increases exponentially, whereas the amount of time to complete an uncoupled design increases linearly. As can be seen, the effects of coupling on design iteration cannot be ignored. For a fully coupled 5x5 matrix, it can take up to 10-20 times longer than an uncoupled design. Therefore, it would be in the designer's best interest to use axiomatic design to create designs that are uncoupled.

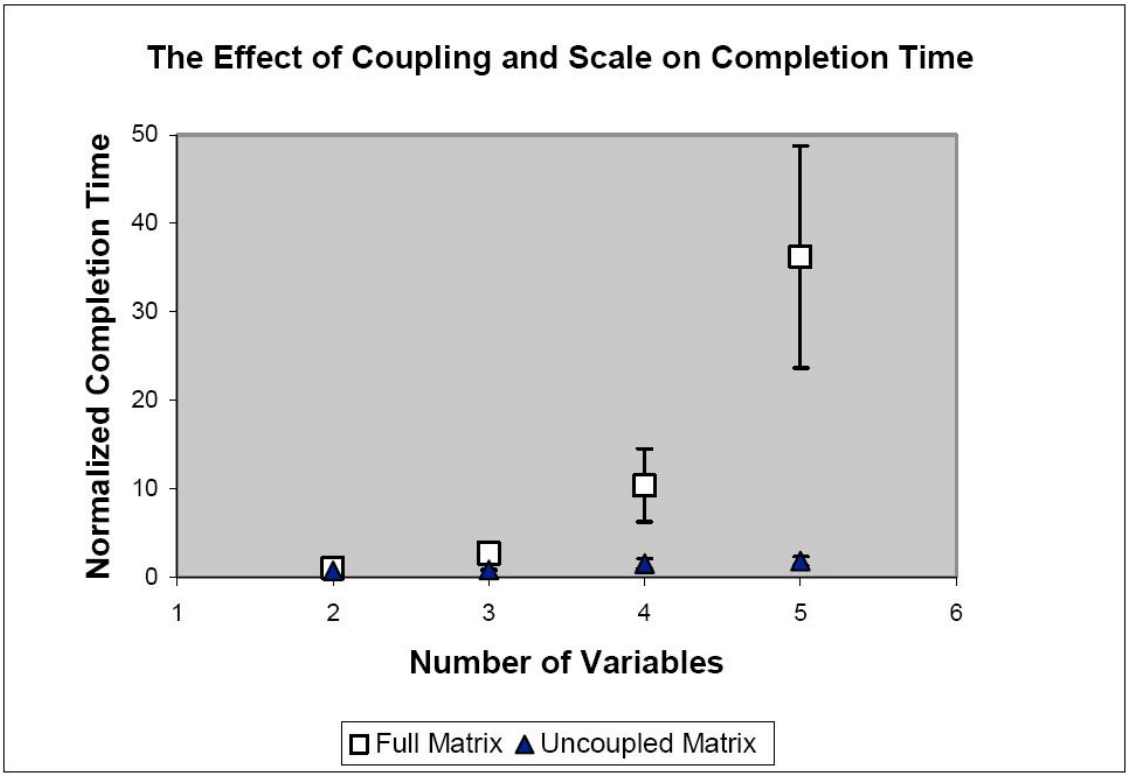

Figure 29: Results of experiments performed by Dan Frey and Nicolas Hirschi. Taken from reference [18].

Imaginary complexity is another contributor to design iteration. Recall from Chapter 4 that time-independent imaginary complexity is a result of the lack of knowledge of a design. Imaginary complexity can be seen by examining the design matrix in Equation 27. The small x in A_{14} indicates a coupling that the designer is unaware of. After having set all FRs, the designer will discover that FR_1 is no longer satisfied. Unless he observed that FR_1 had changed when DP_4 was varied, then he would not know whether the coupling was caused by DP_2, DP_3 or DP_4. Further experimentation would reveal that DP_4 was causing the coupling. The designer would have to iterate between FR_1 and FR_4 until they are both satisfied and then set FR_2 and FR_3. In the process of discovering the small x at A_{14}, the designer would have varied DPs needlessly. If the coupling at A_{14} had been known a priori, then the designer would have resolved the coupling between FR_1 and FR_4, and then set FR_2 and FR_3, thereby minimizing the amount of iteration.

$$\begin{Bmatrix} FR_1 \\ FR_2 \\ FR_3 \\ FR_4 \end{Bmatrix} = \begin{bmatrix} X & 0 & 0 & x \\ X & X & 0 & 0 \\ 0 & X & X & 0 \\ X & 0 & 0 & X \end{bmatrix} \begin{Bmatrix} DP_1 \\ DP_2 \\ DP_3 \\ DP_4 \end{Bmatrix}$$

Equation 27

The above example outlined how imaginary complexity caused unnecessary iteration in a coupled design. It is possible for unnecessary iteration to be performed with a decoupled design as well. If the designer had not used axiomatic design to construct a design matrix, then unnecessary iterations will be performed. Suppose that a fully decoupled 4 FR-DP problem, as in Equation 28, was being solved. This matrix suggests that a unique sequence of DP variations must be performed in order to satisfy all the FRs. FR_1 must be set first, followed by FR_2, FR_3 and then FR_4. If the designer does not follow this sequence, then the system will appear to be coupled. The probability of choosing the correct sequence is 1/24, or 1/(4!). The designer will iterate until the correct sequence is chosen.

$$\begin{Bmatrix} FR_1 \\ FR_2 \\ FR_3 \\ FR_4 \end{Bmatrix} = \begin{bmatrix} X & 0 & 0 & 0 \\ X & X & 0 & 0 \\ X & X & X & 0 \\ X & X & X & X \end{bmatrix} \begin{Bmatrix} DP_1 \\ DP_2 \\ DP_3 \\ DP_4 \end{Bmatrix}$$

Equation 28

In fact, we can determine the expected number of steps given the size of the design matrix. Steps here are defined as the number of times that the FRs are set. For example, two steps would indicate that the designer had set all the FRs once, and then realized he used the wrong sequence. He would have to set all the FRs again, but this time in the correct order. If there is one step, this means that all FRs were set only once. This represents the case when there is no iterations. Suppose the designer decides to try different sequences, never picking the same sequence, until the correct sequence is chosen. Assuming that the FRs and the DPs are already chosen, and the design matrix is only known to be decoupled, then the expected number of steps can be determined. It is analogous to the following probability problem. Suppose that we have a total of m balls in a jar of which k are red balls and m-k are green balls. We pull one ball randomly from the jar. If it is a green ball, we throw it away. If it is a red ball, then we record the number of balls we've picked and stop the game. We want to know the expected number of balls we have to pick up to and including the first red ball picked. The number of total balls, m, is the number of possible sequences, which is n! for an nxn matrix. The number of red balls, k, is the number of correct sequences. The number of green balls, m-k, is the number of incorrect sequences. The expected number of balls we have to pick up to and including the first red ball is the expected number of sequences (or steps) we have to try until we pick the right sequence. The solution to this problem can be seen in Figure 30. The probability of picking the first red ball on the y^{th} try is given in Equation 29. The expected number of steps is given by Equation 30.

$$P(Y = y \mid K = k) = \frac{(m-y)!}{m!} \frac{(m-k)!}{(m-k-y+1)!} \times k$$

Equation 29

$$\mathrm{E}[\mathrm{Y} \mid \mathrm{K}] = \sum_{\mathrm{y}=1}^{m-k+1} y * P(\mathrm{Y} = y \mid K = k)$$

Equation 30

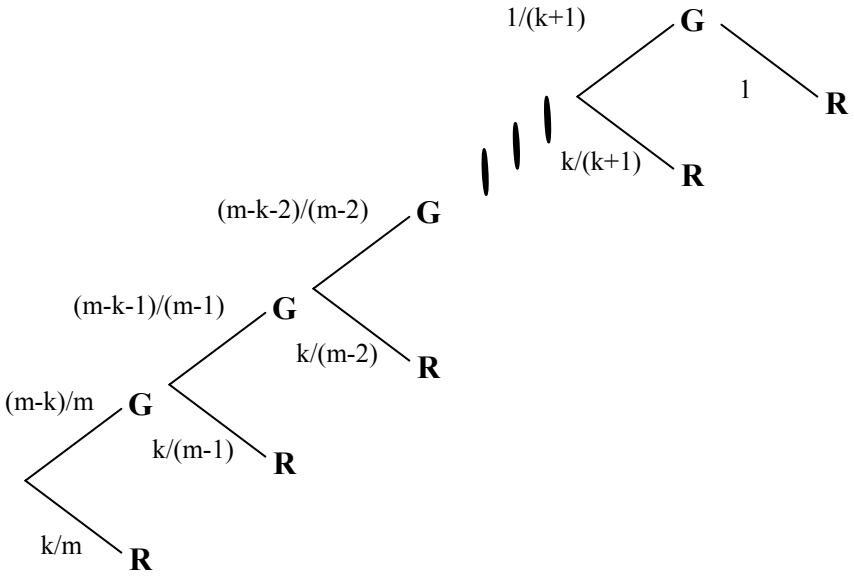

Figure 30: A tree diagram describing the probability of choosing the first red ball from a jar of m balls, k red balls and m-k green balls. R denotes choosing a red ball, while G denotes choosing a green ball.

The expected number of steps is dependent on the number of correct sequences, k, that are available. This dependency is plotted for a 4x4 matrix in Figure 31, where n = 4 and m = 4! = 24. For k = 1, a fully decoupled matrix as in Equation 28, the expected number of steps will be 12.5. For k = 24, a fully uncoupled matrix, the expected number of steps will be 1, meaning that there is no iteration.

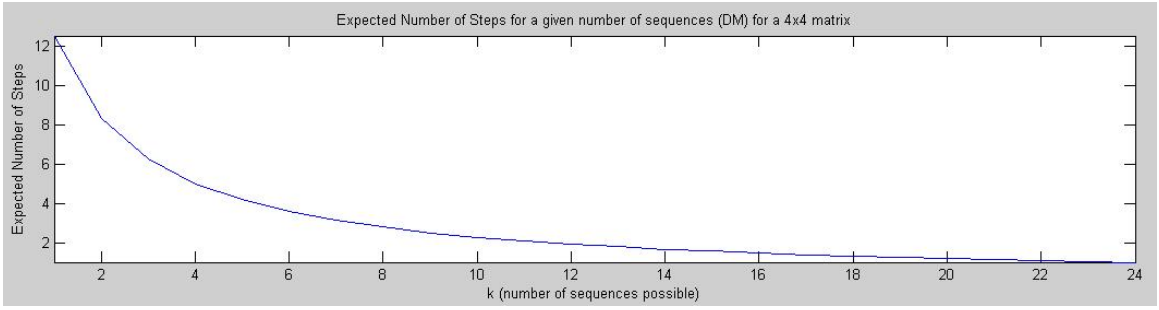

Figure 31: The expected number of iterations for a given number of correct sequences for a 4x4 matrix.

The number of correct sequences, k, is entirely dependent on the amount of decoupling in the design matrix. k = 1 corresponds to a lower or upper triangular matrix, and k = 24 corresponds to a diagonal matrix, while values of k between 1 and 24 correspond to sparsely decoupled matrices.

Suppose that we have a 4x4 FR-DP design problem that we know is decoupled. We are unsure of the severity of the decoupling, and therefore are unsure of the correct number of sequences. If we plan on following the design process of choosing different sequences until the

correct sequence is chosen, then what is the expected number of steps? For a 4x4 matrix, there are $2^6 = 64$ different decoupled matrices because there are 6 off diagonal elements that can be an X or 0. Since we are unsure which of the 64 matrices we have, we must assume that we can have all 64 with equal probability. Of the 64 matrices, the possible number of correct sequences varies. The probability mass function of the number of correct sequences possible for a random decoupled 4x4 matrix can be seen in Figure 32. This graph shows that if a decoupled matrix is chosen at random, then values of k between 1 and 8 are more probable.

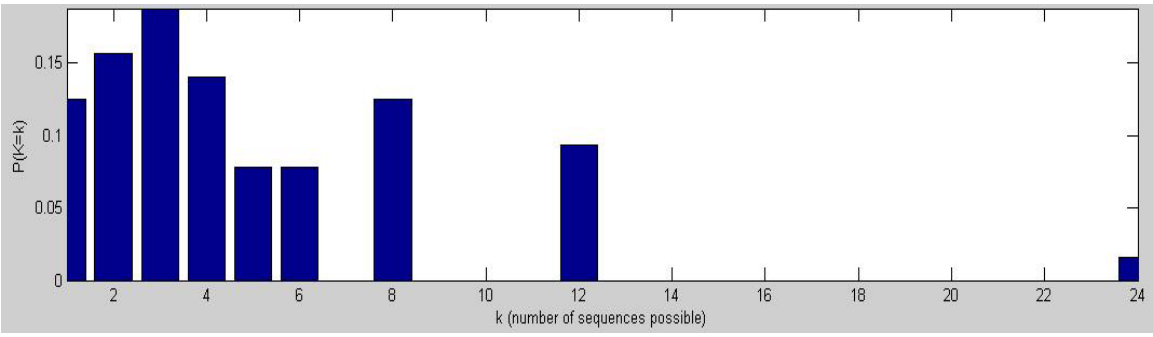

Figure 32: The probability mass function for the number of correct sequences, k, for a random decoupled 4x4 matrix.

By combining Figure 31 and Figure 32, we can determine the probability mass function for the expected number of steps for a random matrix, seen in Figure 33. The expected number of steps for a randomly chosen decoupled 4x4 matrix, as indicated by the vertical blue line, is nearly six. If we had instead written down the design matrix to begin with, we would have known the correct sequence immediately, and would never have had to iterate. Therefore, the cost of not using axiomatic design to find the design matrix and determine the correct sequence to vary DPs for a 4x4 design problem is nearly six times.[18]

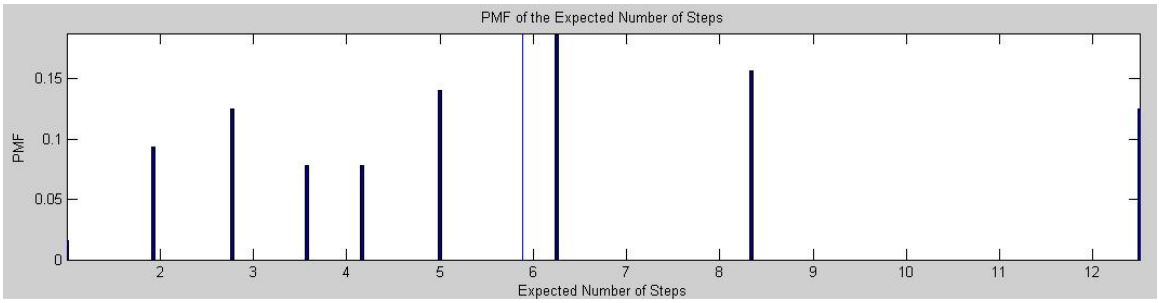

Figure 33: The probability mass function of the expected number of steps for a random decoupled 4x4 matrix.

Design Structure Matrices - Discussion on Design Iteration

The principal feature of design structure matrices is to determine which part of the development process is most iterative, and then apply methods to more effectively deal with that iteration. Identifying the most iterative set of tasks can be done using a method prescribed by Smith and Eppinger.[19] By analyzing the eigenstructure of the work transformation matrix, the aspects of the design process that most contribute to the total work can be identified.

[18] See the appendix for the MATLAB code used to generate these figures.
[19] See reference [3].

Just like in the analysis of dynamic systems, we can identify the behavior of the eigenmodes, also called design modes. The eigenvalues determine the rate of decay of a particular mode. The eigenvector determines the shape of the mode. In terms of the work transformation matrix, we can analyze how the work of each design mode decays in time or how a particular mode is iterating through oscillation. The work transformation matrix A can be broken down into a multiplication of its eigenvalues and eigenvectors, as seen in Equation 31, where S is the matrix of eigenvectors and Λ is the matrix of the corresponding eigenvalues on its diagonal. The total work vector, U, can be calculated from the S, Λ, and the initial work vector (u_0) quickly using Equation 32. The best method of ranking the eigenmodes is by ranking the values of the diagonals of the matrix $(I - \Lambda)^{-1}$. The highest positive real values of the matrix indicate which eigenmodes contributes the most to the total work vector.

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

Equation 31

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

Equation 32

Smith and Eppinger applied this method to identifying the most significant design mode of a brake-system design process at General Motors. They constructed a work transformation matrix, A, by surveying the brake-system design team about the tasks typically required to complete a brake-system and the interactions between those tasks. By determining the most significant eigenmodes of the A matrix, they were able to identify two key problems in GM's brake-system design process: the stopping distance and thermal problems. These two problems consist of a set of tasks that have a highly iterative nature and contribute most to the total work required to complete the brake-system design. By knowing of these problems, GM can attempt to make the iteration process go faster or reducing the total number of iterations by implementing process improvements.

They suggest the following process improvements to make the iteration process go faster:

- Computer-aided design systems which accelerate individual design tasks
- Engineering analysis tools that reduce the need for time-consuming prototype/test cycles.
- Information systems that facilitate the rapid exchange of technical information.

They suggest the following process improvements to lessen the amount of iterations:

- Improve coordination of individuals whose work depends on one another.
- Reduce the team size, to allow individuals to work more efficiently.

Therefore, knowing the most iterative aspects of the design process enable for process improvements that can minimize the cost impact of those highly iterative processes to the overall cost of the development process. These key cost drivers can be readily attained by analyzing the process matrix referred to in Chapter 3 and be used to minimize the cost of development.

Summary

Key cost drivers are essential to the cost minimization effort. Knowing the aspects of the design and design process that most contribute to the cost before the costly development cycle sets in full motion can be very valuable information. In this chapter, two types of key cost drivers were presented: the most expensive FRs and design iteration.

Having knowledge of the most expensive FRs could prove useful in understanding why certain aspects of the design and components are more costly than others. With this knowledge, the designer could seek to eliminate or minimize those costly aspects of the design. Unfortunately,

determining the most expensive FRs is a difficult task. It requires the mapping of the estimated cost from the costing unit domain into the FR-DP domain. This can be done somewhat artificially by apportioning the cost of each CU to the DPs. The DP-CU matrix aids in this process. Once the cost of DPs have been determined, then the cost of each DP must be apportioned to the FRs. This requires knowledge of the relative magnitude of the DPs affect on each FR. The cost of the DP will lie mostly upon the FRs that are most heavily affected by it. Further research can improve upon the method of costing the DPs by either determining their costs individually or by an improved mapping from the CU domain to the DP domain. Also, determining the mapping of the cost of DPs into the FR domain is the natural extension of that research effort.

Design iteration is another key cost driver that impacts the cost of the development cycle. Axiomatic design shows how design iteration can be minimized by controlling the design range and by minimizing coupling and imaginary complexity. By leaving large design ranges, it becomes easier to choose the right DPs to satisfy the FRs. Coupling also adds to design iteration. Depending on the size of the coupling, the amount of design iteration can be amplified greatly, as shown in reference [18]. Imaginary complexity adds to the number of design iterations. By not using axiomatic design to determine the correct sequence in which to vary the DPs, the imaginary complexity involved amplifies the amount of design iteration even further. For a 4x4 decoupled case, it was shown that not using axiomatic design could cost six times more on average. Design structure matrices provide other valuable insight into design iteration. It identifies the most iterative tasks and suggests ways of effectively dealing with the iteration. A brief summary of a method developed by Smith and Eppinger of analyzing the eigenstructure of the work transformation matrix was discussed. By determining the eigenmodes (or design modes) that most significantly contribute to the total work vector, one can determine the set of tasks that will be most problematic. One this has been determined, methods that minimize the number of iterations or speed up the iteration process can be implemented.

Chapter 6

Conclusion

The life-cycle cost is the main goal of every cost estimation effort. By knowing the cost at each phase in the life of a system, the designer and customer can better understand the nature of the product and can use this information along with benefits analysis in order to determine the true worth of the system. Is it worth it to proceed with the design? Can we change the design somehow to reduce production costs? Will it be worth it to the customer to operate this system? All these questions can be tackled effectively by analyzing the life-cycle cost.

Cost estimation is a "black art," consisting of one part science and one part art. Current cost estimation practices rely on physical parameters of the design and correlate these parameters to historical cost data. These physical parameters cannot completely communicate the complexity and difficulty in a design and historical data cannot make up for this fact. Without a proper measure of the complexity of a design, engineers will be unable to communicate the possible problems in the design. Without a proper measure of the complexity of a design, cost estimation will remain unable to estimate the cost of future systems. Cost and design engineers have become comfortable with passing these parameters without fully understanding and communicating the complexity of the design. How can someone justify the cost of a system without even understanding the complexity of the system? The two are clearly correlated. New methods should be developed in order to grapple with the cost estimation of complex systems.

The focus of my research is to understand the problems in design and in the design process that lead to failures in cost estimation. What is it about complex systems that make it so difficult to estimate the cost? Researchers of the Axiomatic design theory have grappled with complexity of a system with some fair amount of success. My research has focused on using what knowledge of complexity that has become available to improve cost estimation. Axiomatic design also serves as a very good medium for communication of a design. Recent research in process design has also led to a better understanding and predictions of its iterative nature. By combining the information from axiomatic design and process design theory, a better measure of cost can be obtained. This information can also be used to determine the key cost drivers of a system.

The goal of this thesis was to:

- Enhance the credibility of cost estimation by creating the cost model based on the FR-DP map.

By basing cost estimation upon the FR-DP design matrix, we can gain a better insight into how the FRs of a system affect the cost. Current cost estimation practices only take into consideration physical parameters, without any regards to whether the FRs even satisfy the DPs. By creating a costing unit domain, cost information that already exists in a similar form as a work breakdown schedule in most companies can be used. Each costing unit can be linked to the set of DPs that define its design by using a DP-CU matrix. Now information from the actual design and cost can easily be communicated. Ideally, one would derive the costing units from the FR-DP map. With such a setup, the scope of cost estimation is defined by the FRs and customer needs of a system. Additionally, traceability information is inherently included in the design.

- Quickly estimate the cost impact of changes introduced to a system.

I focused on determining the cost impact of changes introduced to the development phase. The procedure for determining the cost impact of changes to the development phase is the following: 1) identify which FRs will change due to a change in constraints or customer needs, 2)

determine the DPs affected by that FR change, 3) determine the CUs that will be changed due to the DP changes, 4) determine how the changes to this initial list of CUs will affect other CUs that they physically interact with, and 5) determine the amount of additional development labor required to make changes to those CUs. A CU-CU matrix was created in order to deal with step 4: to identify how the components physically interact with each other. A work transformation matrix was used in step 5 in order to quantify the amount of additional labor that would be incurred due the specified FR changes. When implemented in software, this procedure can instantly estimate the cost impact of changes introduced to a system.

Operations were defined using the axiomatic design framework. Operations are necessarily created due to the combinatorial complexity of FRs. A FR that has combinatorial complexity implies a new operational FR that maximizes the common range. The DP then is to reinitialize the time-variant DPs at a certain frequency. The design of operation then is the design of how to reinitialize those DPs and at what frequency.

The cost of operation is defined by three parameters: frequency of reinitialization, cost of reinitialization, and the cost to maintain the capability of reinitialization. The optimal frequency of reinitialization was found to be the inverse of the time when the cost of reinitialization was the same as the cost of functional failure multiplied by the probability of failure. The cost of functional failure is subjective and depends on the value that the customer places on that function. The task of cost estimation for operation, therefore, is to define all the operations from the FR-DP map and then to determine the values of the three cost parameters for each operation. Once the baseline operation cost is defined, one could predict the cost impact of changes to operation cost by determining what the change is in the cost parameters.

- Identify key cost drivers

Key cost drivers are critical in the cost minimization process. Only by knowing which parts of the design or the design process are most costly, can one attempt to minimize their cost. Two key cost drivers were identified: the most expensive FRs and design iteration.

The most expensive FRs can be determined by mapping the cost from the costing unit domain back into the FR-DP domain. This can be done using the DP-CU matrix by mapping the cost of each costing unit to the appropriate DPs. Then the cost of each DP must be appropriately apportioned to the FRs according to its relative effect on each of the FRs. Design iteration is another key cost driver. Axiomatic design reveals that design iteration is caused by small design ranges, coupling, and imaginary complexity. By using Axiomatic design, one can reduce the amount of design iteration and decrease development costs. Design structure matrices provide yet another way of identifying design iteration. The most iterative sets of tasks in the design process can be determined by examining the design modes of the work transformation matrix, or by examining the sets of tasks that contribute the most time to the design process. This can be done by the eigenstructure analysis developed by Smith and Eppinger and summarized here. Once the most iterative set of tasks are determined, several methods can be applied to reduce the number of iterations or speed up the iterating.

Suggestions for Future Research

There is much more research to be done in terms of cost estimation of the cost impact of a design change and key cost drivers.

The method of determining the amount of time required to make the specified design changes uses a parallel model of iteration, which is not the case in real life but only an approximation. The tasks required for the development phase in real life are completed sequentially, in parallel and in an overlapping manner. In addition, the values of the work transformation matrix are not always time-invariant, as the parallel model of iteration might

suggest. Instead effects like the learning curve could have significant effects. Improved, more sophisticated process models already exist and only need to be applied to this model.[20]

Axiomatic design must address production of a design and a model must be developed that can predict the facets of production costs. The ability to predict the cost impact of a design change to operations cost is also a very hot topic, but also a very daunting task. This would require coming up with a model that would take some information about the type of change to the design, and predict how that would change the cost of reinitialization, the frequency of reinitialization, and the cost of maintaining the capability of reinitialization for each operation.

The ability to determine the cost of FRs must be further investigated in detail to gain better insight as to how the cost of costing units truly relates to the FR-DP mapped. Once this is obtained, key cost drivers will be easy to determine and the proper cost minimization efforts can be employed.

Also of interest would be to further refine the lower and upper bounds of the cost estimate. The lower and upper bounds of development cost are largely determined by the amount of design iteration involved. By monitoring the design ranges, the amount of coupling, and predicting the amount of imaginary complexity, better estimates of these bounds could be obtained.

[20] See reference [5].

Glossary of Terms

% Rework, α	The % Rework (α) is a measure of the magnitude of a change. It is calculated as the number of affected interfaces or attributes from a change out of the total number of interfaces and attributes. This measurement is used to determine the amount of rework that must be done to complete the design change. It indicates that a (1- α)*100% percentage of the work is kept, and α*100% is redone.
Common Range	The common range is the overlapping region of the design and system ranges. One can determine the probability of success of a functional requirement by integrating its probability density function on the common range
Constraint	Constraints (Cs) are bounds on acceptable solutions.
Design Parameter	Designer Parameters (DPs) are the key physical variables (or other equivalent terms in the case of software design, etc.) in the physical domain that characterize the design that satisfies the specified FRs.
Design Range	The design range is the range of values that a functional requirement can have and yet still be satisfied.
Development Cost	The cost associated with performing the activities from the conception to directly before manufacturing the first lot. These activities include design, testing and evaluation and require labor, materials and facilities to be completed. The cost of developing a means of production is considered a development cost. This cost is paid for by the developer of the system.
Functional Requirement	Functional requirements (FRs) are a minimum set of independent requirements that completely characterizes the functional needs of the product (software, organization, system, etc) in the functional domain. By definition, each FR is independent of every other FR at the time the FRs are established.
Life-Cycle Cost	The life-cycle cost is the sum of the development, production and operation costs. Life-cycle cost estimates are important making decisions about whether to proceed with a program or a specific set of changes.
Operations Cost	The cost associated with using a system. The key parameters of operations cost are the cost of reintialization, the frequency of reinitialization, and the cost of maintaining the capability of reinitialization. This cost is paid for by the user of the system.
Phase of Change	Changes are made at different phases of a program. The phase of change in the model measures at what percentage design completion the change is made.
Production Cost	The cost associated with producing all lots of a system. This does not include the cost of developing manufacturing processes, hiring employees, and setting up facilities, which are inherently development costs. This cost is paid for by the producer of the system, which is oftentimes the developer.
reinitialization	The way to transform time-variant combinatorial complexity into periodic complexity is to reinitialize the functional requirement so as to maximize the common range and prevent failure.
System Range	The system range is the range of values that the functional requirement actually has. The system range is usually defined by the span of the probability density function of a functional requirement.
time-dependent combinatorial complexity	DPs that satisfy an FR undergoing combinatorial complexity are changing over time due to degrading mechanisms such as friction, fouling, and fatigue. This decreases the common range over time, thereby increasing the uncertainty over time.
time-dependent periodic complexity	DPs that satisfy an FR are changing over time, but are periodically reinitialized in order to maximize the common range and minimize the uncertainty.
time-independent imaginary complexity	Time-independent imaginary complexity is uncertainty that is the result of the lack of knowledge of a design.
time-independent real complexity	The uncertainty that a functional requirement will fall within the design range.
u0	The initial work vector, u0, represents the initial amount of work required to complete each task assuming completely independent tasks. It is a vector of 1's.
Work Load Vector	The Work Load vector (WL) is the total amount of time required to complete each task, given all the interdependencies.
Work Transformation Matrix	A representation of the interdependencies between tasks. It is used in order to calculate the total amount of work required to complete a certain set of interdependent tasks.

Appendix – Matlab Code

mainScript.m

```
%%%
%%% mainScript.m
%%% Peter Jeziorek
%%% 12/06/2004
%%%
clear;

n = 4;
N = factorial(n);
L = n*(n-1)/2; %number of elements on one side of the diagonal

%%%
%%% Create matrices
%%%

%generate a vector of all the possible off-diagonal elements
sizeOfList = 0;
for i=1:n
   for j=1:n
      if ( j < i ) %if it is a lower-triangular off-diagonal element
         sizeOfList = sizeOfList + 1;
         listOfElements(sizeOfList,1) = i; %add it to the list of elements we care about
         listOfElements(sizeOfList,2) = j;
      end;
   end;
end;

numberOfMatrices = 0;
for i=0:L %Create matrices with i X's inserted into the off-diagonal elements
   %initialize lower triangular off-diagonal elements
   if ( i == 0 )
      %initialize a diagonal matrix
      A = zeros(n,n);
      for j=1:n
         A(j,j) = 1;
      end;

      numberOfMatrices = numberOfMatrices + 1;
      for j=1:n
         for k=1:n
            allMatrices(numberOfMatrices,j,k) = A(j,k);
         end;
      end;
   else
      combinations = combntns([1:L],i);
      [numCombinations,sizeOfCombinations] = size(combinations);
      %create matrices for each type of combination
```

```
      for j=1:numCombinations
         %initialize a diagonal matrix
         A = zeros(n,n);
         for k=1:n
            A(k,k) = 1;
         end;
         for k=1:i
            A(listOfElements(combinations(j,k),2),listOfElements(combinations(j,k),1)) = 1;
         end;

         numberOfMatrices = numberOfMatrices + 1;
         for j=1:n
            for k=1:n
               allMatrices(numberOfMatrices,j,k) = A(j,k);
            end;
         end;
      end;
   end;
end;

fid=fopen('test.dat','w');

khistory = zeros(1,factorial(n));
%calculate k for each matrix and sort all the matrices
for i=1:numberOfMatrices
   %create a usable 2D matrix
   for j=1:n
      for k=1:n
         B(j,k) = allMatrices(i,k,j);
      end;
   end;

   [k, sequences] = determineSequences(B);

   %store k's
   khistory(k) = khistory(k)+1;

   B;
   k;
   sequences;
   %fprintf(fid,'Matrix %i\n',i);
   %fprintf(fid,'%i %i %i\n',B');
   %fprintf(fid,'k = %i\n',k);
   %fprintf(fid,'%i %i %i,',sequences); %this line is messing up somehow.
   %fprintf(fid,'\n\n',0);
end;

khistory

fclose(fid);
```

```
%%%
%%% Create graphs
%%%

k = [1:1:N];

[Ex,varx,stdev] = expectedNumberOfIterations(n);
Ex
varx
stdev
subplot(3,1,1);
plot(k,Ex);
title(['Expected Number of Steps for a given number of sequences (DM) for a
',int2str(n),'x',int2str(n),' matrix']);
xlabel('k (number of sequences possible)');
ylabel('Expected Number of Steps');
axis([min(k) max(k) min(Ex) max(Ex)]);
legend('off');

kprobability = khistory/sum(khistory);
WeightedEx = sum(Ex.*kprobability)
varEx = sum(Ex.^2.*kprobability)-WeightedEx^2
stdDevEx = sqrt(varEx)

subplot(3,1,2);
bar(k,kprobability);
title(['PMF of the number of correct sequences, k, for a random decoupled
',int2str(n),'x',int2str(n),' matrix']);
xlabel('k (number of sequences possible)');
ylabel('P(K=k)');
axis([min(k) max(k) min(kprobability) max(kprobability)]);
legend('off');

subplot(3,1,3);
bar(Ex,kprobability);
line ( [WeightedEx, WeightedEx], [0, 1] ); %average
title('PMF of the Expected Number of Steps');
xlabel('Expected Number of Steps');
ylabel('PMF');
axis([min(Ex) max(Ex) min(kprobability) max(kprobability)]);
legend('off');
```

createPossibleMatrices.m

```
%%%
%%% createPossibleMatrices.m
%%% Peter Jeziorek
%%% 12/06/2004
%%%
```

```
%generate every decoupled lower-triangular combination of nxn matrix
clear;
n = 4;
L = n*(n-1)/2; %number of elements on one side of the diagonal

%generate a vector of all the possible off-diagonal elements
sizeOfList = 0;
for i=1:n
   for j=1:n
      if ( j < i ) %if it is a lower-triangular off-diagonal element
         sizeOfList = sizeOfList + 1;
         listOfElements(sizeOfList,1) = i; %add it to the list of elements we care about
         listOfElements(sizeOfList,2) = j;
      end;
   end;
end;

numberOfMatrices = 0;
for i=0:L %Create matrices with i X's inserted into the off-diagonal elements
   %initialize lower triangular off-diagonal elements
   if ( i == 0 )
      %initialize a diagonal matrix
      A = zeros(n,n);
      for j=1:n
         A(j,j) = 1;
      end;

      numberOfMatrices = numberOfMatrices + 1;
      for j=1:n
         for k=1:n
            allMatrices(numberOfMatrices,j,k) = A(j,k);
         end;
      end;
   else
      combinations = combntns([1:L],i);
      [numCombinations,sizeOfCombinations] = size(combinations);
      %create matrices for each type of combination
      for j=1:numCombinations
         %initialize a diagonal matrix
         A = zeros(n,n);
         for k=1:n
            A(k,k) = 1;
         end;
         for k=1:i
            A(listOfElements(combinations(j,k),2),listOfElements(combinations(j,k),1)) = 1;
         end;

         numberOfMatrices = numberOfMatrices + 1;
         for j=1:n
            for k=1:n
```

```
                allMatrices(numberOfMatrices,j,k) = A(j,k);
            end;
        end;
    end;
end;

fid=fopen('test.dat','w');

khistory = zeros(1,factorial(n));
%calculate k for each matrix and sort all the matrices
for i=1:numberOfMatrices
    %create a usable 2D matrix
    for j=1:n
        for k=1:n
            B(j,k) = allMatrices(i,k,j);
        end;
    end;

    [k, sequences] = determineSequences(B);

    %store k's
    khistory(k) = khistory(k)+1;

    B
    k
    sequences
end;

khistory

fclose(fid);
```

determineSequences.m

```
%%%
%%% determineSequences.m
%%% Peter Jeziorek
%%% 12/06/2004
%%%

function [k,correctSequences] = determineSequences(A)
%Determine the number of sequences given an nxn matrix, A

    nxn = size(A);
    n = nxn(1);
    FR = zeros(n,1);
    numberOfCorrectSequences = 0;
    sequences = perms(1:1:n);
```

```
   %check each sequence
   for k=1:factorial(n);
      %for each DP
      for j=1:n
         %for each FR
         for i=1:n
            if ( (A(i,sequences(k,j)) == 1) & (i == sequences(k,j)) )
               FR(i) = 1;
            elseif ( A(i,sequences(k,j)) == 1 )
               FR(i) = 0;
            end;
         end;
      end;

      returnFlag = 1;
      for i=1:n
         if ( FR(i) == 0 )
            returnFlag = 0;
         end;
      end;

      if ( returnFlag == 1 )
         numberOfCorrectSequences = numberOfCorrectSequences + 1;
         for i=1:n
            correctSequences(numberOfCorrectSequences,i) = sequences(k,i);
         end;
      end;

      k = numberOfCorrectSequences;
   end;
```

expectedNumberOfIterations.m

```
%%%
%%% expectedNumberOfIterations.m
%%% Peter Jeziorek
%%% 12/06/2004
%%%

function [Ex,varx,stdev] = expectedNumberOfIterations(n)

N = factorial(n);
k = [1:1:N];

for J = 1:N
   Esum = 0;
   Esumsquared = 0;
   for I = 1:(N-k(J)+1)
```

```
        %p(I) = factorial(N-k(J))*factorial(N-I)*k(J)/(factorial(N-k(J)-I+1)*factorial(N)); %too
large
        %for following line explanation, see http://www.mit.edu/~pwb/cssm/matlab-
faq.html#factorialratio
        p(I) = exp(gammaln(N-k(J)+1)+gammaln(N-I+1)+log(k(J))-gammaln(N-k(J)-I+1+1)-
gammaln(N+1));
        Esum = Esum + p(I)*I;
        Esumsquared = Esumsquared + p(I)*I^2;
    end;
    Ex(J) = Esum;
    varx(J) = Esumsquared - Esum^2;
    stdev(J) = sqrt(varx(J));
end;
```

References

[1] Suh, Nam Pyo. *Axiomatic Design: Advances and Applications*. Oxford University Press, New York, 2001.

[2] Suh, Nam Pyo. *The Principles of Design*. Oxford University Press, New York, 1990.

[3] Smith, Robert P. and Eppinger, Steven D. "Identifying Controlling Features of Engineering Design Iteration," *Management Science*, Vol. 43, Issue 3, pp. 276-293 1999.

[4] Eppinger, Steven D., Whitney, Daniel E., Smith, Robert P. and Gebala, David A. "A Model-Based Method for Organizing Tasks in Product Development," *Research in Engineering Design*, 1994.

[5] Cho, Soo-Haeng and Eppinger, Steven D. "Product Development Process Modeling using Advanced Simulation," Proceedings of ASME Design Engineering Technical Conference (DETC) 2001.

[6] Isakowitz, Steve J. *NASA Cost Estimating Handbook 2002*.

[7] Lee, Taesik. "Complexity Theory in Axiomatic Design." Cambridge, MA 2003.

[8] Cooper, Kenneth G. "Naval Ship Production: A Claim Settled and a Framework Built." *Interfaces*, Vol. 10, No. 6, pp. 20-36 December 1980.

[9] Central Artery/Tunnel Project. http://www.bigdig.com/. Published on WWW; accessed in 2004.

[10] Axtman, Kris. "In Boston, a 'Big Dig' into taxpayer pockets." http://search.csmonitor.com/durable/2000/02/18/p3s1.htm. Published on WWW; accessed in 2004.

[11] Bush, George W. President Bush Announces New Vision for Space Exploration Program http://www.whitehouse.gov/news/releases/2004/01/20040114-3.html. Published on WWW; accessed in 2004.

[12] Wade, Mark. Encyclopedia Astronautica: The Shuttle http://www.astronautix.com/lvfam/shuttle.htm. Published on WWW; accessed in 2004.

[13] Bertsekas, Dimitri P. and Tsitsiklis, John N. *Introduction to Probability*. Athena Scientific, Belmont, Massachusetts, 2002.

[14] Suh, Nam Pyo, *Complexity: Theory and Applications*. Oxford University Press, New York, 2005.

[15] Lurie, David. "Human Space Flight: Fiscal Year 2002 Estimates." http://ifmp.nasa.gov/codeb/budget2002/06_space_shuttle.pdf. Published on WWW; accessed in 2004.

[16] Reaves, Will. "Reusable Launch Vehicle (RLV) Power Generation Systems - Trail from 1st to 2nd Generation." Lockheed Martin Technical Operations.

[17] Futron Corporation. "Space Transportation Costs: Trends in Price Per Pound to Orbit 1990-2000." http://www.futron.com/pdf/FutronLaunchCostWP.pdf. Published on WWW, 2002; accessed in 2004.

[18] Frey, Daniel D. and Hirschi, Nicolas W. "The Effects of Coupling on Parameter Design." Proceedings of the 2002 International Conference on Axiomatic Design.

[19] Shishko, Robert. *NASA Systems Engineering Handbook*. National Aeronautics and Space Administration, Washington D.C., 1995.