Reconfigurable Variation Risk Management Database Tool

by

Gisele M. Proulx

S.B. Mechanical Engineering
Massachusetts Institute of Technology, 1998

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

JUNE 2000

© 2000 Massachusetts Institute of Technology
All rights reserved.

Signature of Author.................................................................................................

Department of Mechanical Engineering
May 22, 2000

Certified by.................................................................

Anna Thornton
Assistant Professor of Mechanical Engineering
Thesis Supervisor

Accepted by.................................................................

Professor of Mechanical Engineering
Chairman, Committee for Graduate Students
Reconfigurable Variation Risk Management Database Tool

by

Gisele M. Proulx

Submitted to the Department of Mechanical Engineering
on May 22, 2000 in partial fulfillment of the
requirements for the Degree of Master of Science

ABSTRACT

Companies have recognized the need to implement variation risk management (VRM) practices to reduce cost, improve quality, and reduce design cycle time. However, poor tools have hindered systematic implementation of the VRM process. Existing tools are not aligned with current VRM practices, have poor interfaces, and are not integrated with other tools or sources of information.

The thesis presents a design for a VRM software tool, and an underlying data structure to address these concerns. The author hypothesizes that a common, underlying data structure can describe all the information from and about the VRM process, and that a flexible tool that is consistent with existing VRM techniques and can be used to enter and manipulate that data.

The tool design uses a layered software approach and the Model-View-Controller (MVC) architecture to implement the data structure and the various software components that interact with it. The more fundamental layer, the data structure, captures about a product and its variation. The data structure is stored in a database as a series of tables. The next layer, the basic functions, interacts with the database and provides a buffer to maintain database integrity. The VRM process is described as a set of process elements (the specific VRM techniques) in a process description that tells what order the elements are performed and by whom. A user interface provides users with a way to interact with the VRM tool.

The data structure was implemented and tried out with data from a partner company. The basic elements of the tool were implemented in software and shown to work properly with the database. The design of the remaining elements of the tool builds on this foundation.

Thesis Supervisor: Anna Thornton

Title: Assistant Professor of Mechanical Engineering
Acknowledgements

I owe great thanks to Professor Anna Thornton. You’ve been an inspiration, a mentor, and a patient advisor. Without your support, guidance, and feedback, none of this would have been possible. My thanks, also, to all the industry people who provided insight and suggestions on my work. Discussions with you always provided renewed inspiration for the work. Special thanks goes to those people whom I interviewed and met with at length in order to understand the problem better, especially Charlie and Ron. I thank the Center for Innovation in Product Development at MIT for sponsoring this research (NSF cooperative agreement Number EEC-9529140) and for providing a home for my work. Thanks to all those at the center, but especially to my officemates who always made it fun to come in the office.

Finally, I owe an incredible debt to my friends and family who encouraged supported me through all this. Thanks to all those who helped and advised me on computer problems I ran into, especially David Wang, without whom I never would have finished. Thanks to all my friends who reminded me and showed me that I could do it, but especially to Matt Braun, Jeremy Brown, Liana Lareau, Erik Nygren, Cindy Dernay Tervalon, and Ingrid Ulbrich. Thanks to my family, for always being there and always believing in me. Lastly, a very special thank you to James Kretchmar. You’ve always been there for me and helped me believe I could make it, even when I didn’t think I would. Thank you for everything.
# Table of Contents

**ABSTRACT** .................................................................................................................................................. 3

**ACKNOWLEDGEMENTS** .......................................................................................................................... 5

**TABLE OF CONTENTS** .................................................................................................................................. 7

**LIST OF FIGURES** ....................................................................................................................................... 10

1 **INTRODUCTION** ....................................................................................................................................... 11

1.1 **OBJECTIVES** ....................................................................................................................................... 14

1.2 **OUTLINE OF THESIS** .......................................................................................................................... 15

2 **VRM TOOL REQUIREMENTS** .................................................................................................................... 17

2.1 **VARIATION RISK MANAGEMENT** .......................................................................................................... 17

2.1.1 **What is VRM?** .................................................................................................................................... 17

2.1.2 **What are KCs?** .................................................................................................................................... 19

2.2 **WHAT IS THE VRM PROCESS?** .............................................................................................................. 21

2.2.1 **Identification** ...................................................................................................................................... 21

2.2.2 **Assessment** ...................................................................................................................................... 22

2.2.3 **Mitigation** ....................................................................................................................................... 23

2.2.4 **Communication and Documentation** .............................................................................................. 23

2.3 **EXISTING TOOLS** .................................................................................................................................... 24

2.4 **PRIOR RESEARCH** ..................................................................................................................................... 27

2.4.1 **Process Capability Database Research** .................................................................................................. 31

2.4.2 **Summary of Research-Based Requirements** ...................................................................................... 32

2.5 **INTERVIEWS** .......................................................................................................................................... 32

2.5.1 **Purpose of interviews** .......................................................................................................................... 33

2.5.2 **Interviewees** ...................................................................................................................................... 34

2.5.3 **Summary of observations from interviews** ......................................................................................... 36

2.6 **REQUIREMENTS** ......................................................................................................................................... 38

2.6.1 **Consistency with Existing Practice** ..................................................................................................... 38

2.6.2 **Platform Independence** ...................................................................................................................... 40

2.6.3 **Information Control** .......................................................................................................................... 41

2.6.4 **Date Tracking** ................................................................................................................................... 42

2.6.5 **System Level Views** ............................................................................................................................ 43

2.6.6 **Data Consistency** .................................................................................................................................. 44

2.6.7 **User Interface** .................................................................................................................................... 45

2.7 **LIST OF REQUIREMENTS** .......................................................................................................................... 45
### 3 INFORMATION ARCHITECTURE

3.1 PROBLEM SCOPE .......................................................................................................................... 48

3.2 SOFTWARE TECHNIQUES ............................................................................................................. 50

3.2.1 The Model-View-Controller Model .......................................................................................... 51

3.2.2 Software Layering .................................................................................................................... 53

3.2.3 Software Layers ....................................................................................................................... 56

3.3 DATA STRUCTURE ELEMENTS ..................................................................................................... 59

3.3.1 Product Architecture ............................................................................................................. 60

3.3.2 Key Characteristic Architecture .......................................................................................... 62

3.3.3 People ..................................................................................................................................... 64

3.3.4 Budgets ................................................................................................................................... 65

3.3.5 Customer Requirements and Environmental Factors ............................................................ 66

3.3.6 Company Specific Elements ............................................................................................... 67

3.4 SUMMARY ..................................................................................................................................... 68

### 4 VRM TOOL DESIGN .......................................................................................................................... 69

4.1 BASIC FUNCTIONS ......................................................................................................................... 70

4.1.1 Database Connectivity ............................................................................................................. 70

4.1.2 Adding, Changing, Deleting, and Viewing Elements .............................................................. 71

4.1.3 Data Caching .......................................................................................................................... 73

4.1.4 Data Verification ..................................................................................................................... 73

4.2 PROCESS ELEMENTS ...................................................................................................................... 75

4.2.1 Search and Sorts ...................................................................................................................... 76

4.2.2 Data Type Specific Elements .................................................................................................. 79

4.2.3 VRM Techniques ................................................................................................................... 80

4.2.4 Reconfigurability .................................................................................................................... 89

4.3 PROCESS DESCRIPTION .................................................................................................................. 90

4.4 USER INTERFACE .......................................................................................................................... 94

4.4.1 Basic User Interface Elements ............................................................................................... 94

4.4.2 Additional Capabilities .......................................................................................................... 96

4.5 ADMINISTRATIVE CONTROLS .................................................................................................... 101

4.6 SOFTWARE ISSUES ....................................................................................................................... 101

4.7 CONCLUSIONS ............................................................................................................................. 103

### 5 CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK ......................................................... 104

5.1 MORE TESTING OF THE DATA STRUCTURE ............................................................................ 105

5.2 EXPLORE USABILITY ACROSS INDUSTRIES ........................................................................... 106

5.3 SUPPORT FOR ELECTRICAL AND SOFTWARE DESIGNS ......................................................... 106
5.4 USER INTERFACES ........................................................................................................... 107
5.5 VISUALIZATION TECHNIQUES FOR SYSTEM LEVEL VIEWS ...................................... 107
5.6 CLOSING REMARKS ...................................................................................................... 108

REFERENCES ..................................................................................................................... 111
List of Figures

FIGURE 1: KC FLOWDOWN ........................................................................................................ 20
FIGURE 2: MODELING COMPLEXITY VS. BREDTH ............................................................... 30
FIGURE 3: MODEL-VIEW-CONTROLLER SCHEMATIC .......................................................... 52
FIGURE 4: ENCAPSULATED DATA ......................................................................................... 54
FIGURE 5: SYSTEM ARCHITECTURE .................................................................................... 56
FIGURE 6: PRODUCT ARCHITECTURE .................................................................................. 60
FIGURE 7: SIMPLIFIED AIRPLANE PRODUCT ARCHITECTURE ......................................... 61
FIGURE 8: KEY CHARACTERISTIC FLOWDOWN ................................................................... 62
FIGURE 9: KC FLOWDOWN WITH SENSITIVITIES AND MULTIPLE PARENTS AND CHILDREN ..... 63
FIGURE 10: AN EXAMPLE OF A BUDGET ............................................................................ 65
FIGURE 11: I-O-C CHART ...................................................................................................... 81
FIGURE 12: BASIC FLOWDOWN VIEW ................................................................................. 83
FIGURE 13: KC FLOWDOWN DETAILS PANEL ..................................................................... 85
FIGURE 14: SAMPLE DISPLAY PARAMETERS WINDOW ....................................................... 87
FIGURE 15: PARAMETER CHOICE, SHOW FAMILY ............................................................... 87
FIGURE 16: EXAMPLE OF SHRINKING FLOWDOWN AREAS ............................................. 88
FIGURE 17: SAMPLE DSM .................................................................................................... 93
FIGURE 18: STARTING FLOWDOWN WIZARD PROMPT ....................................................... 98
FIGURE 19: CHILD ENTRY WIZARD PROMPT ..................................................................... 99

List of Tables

TABLE 1: KC VALIDATION BASED ON COST AND PROBABILITY OF VARIATION IMPACT ... 22
TABLE 2: RESEARCH-BASED REQUIREMENTS FOR VRM TOOL ...................................... 32
1 Introduction

Aerospace, automotive, and consumer product design and manufacturing firms face new challenges in product development; they must develop more complex products faster and at lower costs, while simultaneously increasing quality. (Thornton 1999b) One way to control quality is by managing product variation. Product variation is the “deviation from nominal design values or operational specifications” (Jay 1998) that occurs during manufacturing. Product variation increases cost and decreases quality. Product returns, repairs, warranty costs, rework, scrapped parts, production disruptions, increased inspection, under-utilization of manufacturing capability, customer dissatisfaction, and production ramp-up delays are all caused by variation and cost companies lots of money. Variation is unavoidable, and it is not practical or cost effective to eliminate it completely. Therefore, variation must be managed to produce the best quality product for the lowest cost. Companies must implement systematic ways to address variation at the optimal points. (Thornton 1999a)

Variation risk management, VRM, uses information about variation, and its associated risk and cost, to make informed design decisions, reducing costs and increasing quality. (Thornton, Donnelly, Ertan 1999, Jay 1998) The probability of variation is multiplied by the cost of that variation to give an effectiveness measure. (Thornton 1999a) Teams must identify where the cost of controlling variation will yield the greatest returns based on the effectiveness measure. Companies have been very good about looking at variation on a part level, but often do not address system-level variation successfully (Ardayfio 1998).
Flowdown is one way to view this information. It shows the relationship between parts, subsystems, systems, and customer requirements. Flowdown is integral to the VRM process and to a complete understanding of the system level aspects of a product. Despite this, many companies do not systematically implement or use flowdown techniques (Thornton, Donnelly, Ertan 1999). One reason for this is a lack of appropriate tools for visualizing and accessing the system level impact of variation (Ardayfio 1998). New computer-based tools could allow teams to record and view system level views.

Companies have changed how they use computers in the product development processes. They use computers to keep track of inventory, manage and store information about manufacturing and variation in parts, design products, and document decisions. Partner companies noted that they are doing more and more work on computers. They are trying to achieve “paperless offices” and capture entire designs using computer-aided design (CAD) and other tools. More recently, some of these companies have adopted product information management (PIM) systems for coordinating and integrating a wider range of information.

PIM systems are an attempt to integrate the information about product development to increase knowledge about the product and the development process. They take information about the design from CAD systems and link it to manufacturing information, online design notebooks, and results of computer analysis to document design intent and design choices. Still, discussions with companies show that many PIM
systems only create links between existing pieces of information instead of actually integrating that information to create a richer body of knowledge.

The information from product development is stored now, but companies do not utilize it fully because they do not have the tools to do so (Ardayfio 1998). Information that should or could be shared is not, and people can not find the information they need, or it is not in a format they can use; information may be out of date, but there may not be a way to tell (Tata and Thornton 1999). Appropriate computer tools can provide a way to manage the large amounts of information resulting from product development, including managing risk and variation. One way to manage this information is using VRM databases.

Most companies have databases of their product VRM information, yet few companies actually utilize them. Databases are not integrated with other systems, they do not contain needed analytic capabilities, they are not used across entire organizations, and they are reinvented for many projects. Because of these shortcomings, current databases are not useful within their organizations. (Thornton, Donnelly, Ertan 1999)

Additionally, the user interfaces of the databases are ineffective and difficult to use. They are incompatible with the process and techniques that designers use in their work, and so extra work is needed to "translate" information before being able to input it. The textual, tabular format makes it hard to enter data, and even harder to find data once it is in the database. (Thornton, Donnelly, Ertan 1999) The result, in most cases, is a primarily empty database with questionable data at best. Additionally, the data in the database
cannot be viewed or outputted in useful formats, so engineers have even less incentive to go to the extra work of entering data. (Tata and Thornton 1999) A tool that successfully integrates all the VRM knowledge and presents it in a way consistent with current techniques would be a great improvement. This thesis proposes a tool to address these concerns and provide a way for teams to implement the VRM process as efficiently and easily as possible.

1.1 Objectives

The author hypothesizes that there is an underlying data structure common across companies and industries that defines VRM information, and that a flexible software tool that is consistent with existing VRM techniques can be used to enter and manipulate that data. The different tools companies currently use are simply different ways of viewing and manipulating this data. Companies use many techniques to view data, but they also find new techniques for looking at this data, and need a tool that is flexible enough to incorporate such changes. The system architecture of the VRM tool must provide a robust, reconfigurable way to access, view, enter, and manipulate data. The user interface must be intuitive and consistent with current VRM practices. A good user interface will encourage users to enter information, and as a result, there will be more and better information available for further development. Engineers and managers need to be able to look at information on a system level as well as in detail while the information is still fresh. This added and improved information, along with more ways to view it, will speed up and improve the product development process as decisions can be made more quickly and with better information.
The thesis proposes a design for a VRM tool and the underlying data structure. The data structure was developed and then tested using data from a partner company to see if it accurately represents their VRM process. The basic components of the VRM tool were implemented and shown to work with the underlying data structure. The design of the tool shows how the user interface and higher level elements build off the basic, implemented components.

1.2 Outline of Thesis

This thesis presents the variation risk management process and proposes a software tool to guide users through the process and manage the information resulting from it. The following chapters present the tool design, and the rationale behind it.

Chapter 2, “VRM Tool Requirements,” presents an overview of variation risk management (VRM) and the VRM process. It details how prior research in VRM and interviews at partner companies generated a list of requirements for the VRM tool.

Chapter 3, “Information Architecture,” presents the overall structure of the information in the VRM tool. There is a discussion on the overall design of the tool, followed by an in depth presentation of the common data structure on which the tool is based.

Chapter 4, “VRM Tool Design,” outlines the VRM tool software that interacts with the data structure in Chapter 3.
Chapter 5, “Conclusion,” reviews the findings of the thesis and explores areas for further investigation.
2 VRM Tool Requirements

The reconfigurable variation risk management tool provides a way to implement the variation risk management (VRM) process in a consolidated software environment. A reconfigurable tool can be configured and modified easily to be and remain consistent with the VRM process as practiced within an organization. Each organization can adapt the tool to its needs, and reconfigure the tool when those needs change. Since all team members will be able to use the same tool, they will be able to communicate more quickly and clearly. Through increased team communication and alignment with the VRM process, the tool will aid, not hinder, product development efforts and allow teams to improve quality while minimizing costs. This chapter explains variation risk management and the VRM process, and then shows the requirements for a tool based on this process.

2.1 Variation Risk Management

2.1.1 What is VRM?
Variation risk management (VRM) is a three-step process of continually “identifying, assessing, and mitigating risk associated with variation sensitive product and process parameters.” (Jay 1998) Variation risk is the cost of variation times the probability that manufacturing processes will effect final product quality. This iterative process identifies potential risks, assesses their relative risks, and then reduces risk through carefully chooses mitigation strategies. (Thornton, Donnelly, Ertan 1999)

The goal of VRM is to reduce the cost of producing a product while simultaneously improving quality. If performed correctly, it can also reduce development cycle times by
improving communication and understanding of the product as a whole. However, despite industry’s understanding of the importance of VRM, many companies do not apply VRM until late in the product development when the product is about to be manufactured (Thornton, Donnelly, Ertan 1999). By this point, VRM can have only a minor impact. Many of the more effective, less costly mitigation strategies will not longer be appropriate, and the cost of making changes increases greatly the closer to production the product gets.

VRM focuses attention on those places where design efforts will have the most effect. Variation can be designed out of the product where it will cause problems, while areas where variation will not have an impact on product quality can be left alone. This produces a more robust design that is easier to manufacture because less inspection is needed, and it is only used where absolutely needed.

The VRM process also promotes communication and understanding. The process causes design and manufacturing groups to communicate and gives them a framework in which to do so. People can understand how features on their part impact features on other parts. By promoting the sharing of information, documentation of VRM activities provides even more ways to increase communication and reuse of design elements in future products.

Variation risk management uses key characteristics (KCs) to identify features that impact cost, performance, or safety. Design and process control efforts are then focused on those
features designated as KCs. The following two sections describe KCs and the VRM process.

2.1.2 What are KCs?

Key characteristics [KCs] are the product, sub-assembly, part, and process features that significantly impact the final cost, performance, or safety of a product when KCs vary from the nominal. Special control should be applied to those KCs where the cost of variation is greater than the cost of control.

(Thornton 1999a)

Key characteristics (KCs), also sometimes referred to as Critical Parameters, Critical Specifications, or Special Characteristics, are the basis for VRM. The VRM research group at MIT, headed by Prof. Anna Thornton, uses the above definition for KCs. Every company uses its own internal definition that varies slightly from the one above. Industry members have spent many hours trying to come to agreement on a definition at the annual VRM/KC Symposium. This thesis will use the definition above when referring to KCs.

KCs are linked to each other to form a KC flowdown. A KC flowdown is the hierarchy of part and process features that contribute to the variation of variation-sensitive product requirements. The KC flowdown provides a system view of variation risk and provides a way of understanding the interactions between system elements. The flowdown is then used in the VRM process to assess appropriate places for mitigation strategies.

The Figure 1 shows the flowdown for the customer perception of a car door. The customer perception is the key customer requirement. Three product KCs, seam evenness, the steps between panels, and door closing force, influence the customer’s
perception. The subsystem KCs contribute to the product KCs, and the part and process KCs contribute to the subsystem KCs. (Thornton 1999b)

The KC flowdown is used to identify which features should be controlled and where process improvements will have the most benefit. This identification is not possible without the traceability between the different levels of KCs.

Figure 1: KC Flowdown (Thornton 1999b)
2.2 What is the VRM Process?

The VRM process is the series of three steps used to identify, assess, and mitigate the risk and cost of variation. The entire process is well documented by Don Jay (Jay 1998) in his masters thesis. In addition to the three main steps, there are supporting activities throughout the entire process, documentation and managerial support. Together, these steps and supporting activities provide a framework in which organizations can improve quality, reduce costs, and increase communication, the speed of development, and employees’ understanding of the product. The process, as described by Don Jay, is summarized below.

2.2.1 Identification
During identification, teams select a set of potential KCs for consideration during assessment and mitigation. Formal definitions are needed so all parties agree on how KCs are chosen and what their implications are. Identification should always be driven by high level product and customer requirements. A KC flowdown is used to trace high level requirements to lower level parameters where variation can be controlled if needed. In some cases, other KCs may be defined at intermediate levels. If possible, KCs should be identified as early in the product development process as possible. Early identification will allow problems to be resolved before or early on in implementation. However the KC flowdown is performed, traceability of the KC information must be maintained. This flowdown will allow for an accurate assessment of the impact of variation at lower levels on high level requirements.
2.2.2 Assessment
The assessment stage determines the expected cost of variation based on the probability of undesired parameter variation. Based on the assessment, certain potential KCs are validated for mitigation. Assessment determines which KCs merit allocation of resources for mitigation. The supporting information from the assessment decisions is passed to the mitigation stage. A chart of how validation occurs is shown in Table 1.

<table>
<thead>
<tr>
<th>Cost of Variation</th>
<th>Probability of Variation Impact</th>
<th>Low</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Not a Valid KC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>High</td>
<td>Assessment Required to Validate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>High</td>
<td>Assessment Required to Validate</td>
<td></td>
<td>Valid KC</td>
</tr>
</tbody>
</table>

Table 1: KC Validation Based on Cost and Probability of Variation Impact (Jay 1998)

Assessment must occur early in the product development process to have the most benefit. Often, however, information is not available when desired. Making a poor assessment can be costly, but so can waiting until accurate information is available and possible mitigation strategies are no longer valid. Assessment must balance the cost and risk of early completion with delayed assessment. When possible, assessments should use quantitative instead of qualitative information. Qualitative data is readily available, but quantitative data provides the most accurate assessment. Based on the assessment, a mitigation strategy can be chosen.
2.2.3 Mitigation
Mitigation selects cost-effective alternatives to minimize the risk and cost identified during the previous two phases. Design and manufacturing, and any other impacted parties, communicated to determine what tradeoffs to make. The earlier mitigation occurs, the less costly the alternatives will be. Early in design, a flaw may cost $1000 to redesign, while in production, it could cost over $10 million. Mitigation is only successful when it is based on accurate, current information. Documentation and communication make this possible.

2.2.4 Communication and Documentation
Communication and documentation are crucial to the VRM process. Without them, assessment and mitigation can not occur. Documentation provides well-defined, consistent instructions for the VRM process and provides a way of recording VRM information of that it will be interpreted consistently and available to those who need it when they need it. The VRM tool provides a way to do all these things. With the proper management support, such tools can provide teams with a way to implement their VRM activities starting at the beginning of the product development process.

Current VRM practice uses many different techniques. Each of these techniques uses a different, independent, unlinked software tool. Information in the VRM process is caught in these isolated tools, instead of being shared in a common repository. In cases where organizations have tried to create such repositories, they have failed. Issues such as poor user interface design, lack of interoperability, non-existent or out-of-date data, and discontinuities with the VRM process have doomed these attempts. This rest of this
chapter explores the requirements for a VRM tool which fits with the VRM process instead of creating work, links together the various VRM techniques, and provides a user interface which promotes, instead of hinders, use of the tool. First, a review of existing tools shows the wide range of VRM activities and the tools that support them. Next discussions on prior research and interviews at a partner company provide background about the problems faced in industry and the requirements for a tool to address them. A set of requirements for a VRM tool is included at the end of the chapter.

2.3 Existing Tools

Organizations use many different tools in their current VRM practices. The tools support different stages of the VRM process. Some of these tools are highly specialized for one task, while others are have more general applications. For example, QFD might be used early in the design process to flow high-level customer requirements to designers, while process capability data will be used later to determine appropriate tolerances on parts. Ideally, a new VRM tool should include elements of some of the existing tools and link to or interoperate with others. Industry interviews and literature reviews by the MIT VRM research group resulted in this list of tools.

**Analysis of Variation (ANOVA)** is a statistical package used with design of experiments (DOE), robust design, sensitivity analysis, and related testing (Jay 1998).

**Computer Aided Design/Manufacturing (CAD/CAM)** systems allow engineers to create drawings and virtual 3-D models of part geometries. They do not provide ways to capture functional relationships or flowdown. CAD/CAM systems do link to, and are
linked to from other tools. Usually, these linkages are just references to a particular
drawing, without easy ways to get the information desired from that drawing.

**Design notebooks** are used to document the design process. Design decisions are
captured, but the information is not managed so that it can be retrieved easily.

**Excel® (and other) spreadsheets** are used to keep lists of KCs and other VRM
information. Usually, each engineer will create their own spreadsheet for their part of
design. This means that sharing information with others to create a system view is
difficult. Spreadsheets must be merged by hand, and by the time this is done, the
information can be out of date. Engineers like the ability to format the information in a
way that makes sense to them, but most complain that it is hard to view complex
relationships in a spreadsheet format.

**Expert systems** are “intelligent” software tools that use and learn rules to propose design
solutions. Computer scientists are still studying how to use and create expert systems.
Current expert systems are used for routine problems where the rules are easily
understood.

**Failure Modes and Effects Analysis (FMEA)** are used to evaluate and capture potential
KCs and to help determine causes of failures. FMEAs are not linked, so flowdown can
not be performed with the information.
Input/Output/Constraint (I-O-C) charts are used to view the input, output, and constraint parameters for a module. Linkages between the different I-O-C charts are rarely captured, and when they are, flowdown views are not available, so it is difficult to understand the linkages.

KC databases have been developed at companies to capture flowdown, process capability, and other VRM information. These databases are not integrated with other information systems, do not have analytic or computational capabilities, are difficult to use, and are not used across entire organizations (Thornton, Donnelly, Ertan 1999).

KCTool is a prototype tool for managing KC flowdown developed by the VRM research group. It provides some analytic capabilities, but does not guide the user through decision making and only provides one view of the KC system. The KC information is not linked to other tools.

Process Capability Databases (PCDBs) are used to record and access statistical process capability (SPC) data gathered from production. In theory, this information would be fed back into future designs, but research shows this is not the case because of the way PCDBs are implemented (Tata and Thornton 1999). PCDBs do not interoperate with existing software tools that could use their information.

Product Data/Information Management (PDM/PIM) Systems are used to archive data from the product development process. The scale of these tools varies from simply
gathering comments linked to CAD files to more complex tools statically linking together all the documents supporting design decisions. These tools generally act as a repository for information, without providing any real tools with which to use that information.

**VSA (Variation Systems Analysis)** uses Monte-Carlo simulation to model variation. It is one of the few software tools more managing variation data.

Other specialized tools exist within companies. These tools are designed specifically for the organization using them. Usually, these have a specific narrow focus and they are not linked well with other existing tools. Because these tools are proprietary, they are not included in the list above.

### 2.4 Prior Research

Prior work by the MIT VRM research group and others provides much insight into the problems with current VRM practice and the tools that support it. Research by Professor Thornton provides new ways of thinking about and analyzing Key Characteristics and identifies problems to be addressed by the VRM tool. These include intergroup barriers, lack of systematic flowdown, use of qualitative instead of quantitative data, lack of tools aligned with the VRM process and integration with other tools and techniques, and the need for earlier implementation of the VRM process and for better documentation. Work by Tata (Tata and Thornton 1999) discusses the use of process capability databases (PCDBs). Interviews in industry show that the problems with PCDBs are very similar to those encountered with other VRM tools. A paper by Thornton, Donnelly, and Ertan
includes an extensive review of other work in the field. (Thornton, Donnelly, and Ertan 1999)

Barriers between groups and between organizations and suppliers create many problems. Delays in information transfer can slow the product development process (Tata and Thornton 1999). Lack of communication can prevent customer requirements from being connected well to engineering decisions (Hauser and Clausing 1988). Finally, lack of trust can cause designers not to use information available to them in existing databases (Tata and Thornton 1999). Proper techniques, such as QFD, can promote teamwork and reduce functional barriers, but these techniques can be hard to implement (Hauser and Clausing 1988). Additionally, all groups must have access to the tool and the VRM information. Often, information is only available to certain people and it is lost as a result (Thornton, Donnelly, and Ertan 1999).

All the research by the VRM Group at MIT shows that systematic flowdown of requirements does not happen. At some companies flowdown does not happen at all, and at others it only happens sporadically and in certain cases. Without a complete flowdown, teams can not optimize their designs and strategies over the whole system or product; they can only find local optimizations. It is important to maintain the traceability of KC information so that people can understand the impact of their design choices on others (Jay 1998, Ardayfio 1998).
Currently, qualitative analysis is used to set customer requirements, and in other situations where quantitative methods would provide better results. In some cases this is because quantitative data does not exist or is hard to find, but in others it is because quantitative tools do not exist (Thornton, Donnelly, and Ertan 1999). A new mathematical framework allows teams to predict final product variation from part and process variation and to allocate resources based on a quantitative effectiveness measure, even in cases where only qualitative data is available (Thornton 1999a). Analytic and computational capabilities like these must be included in VRM tools so that companies can use the many quantitative methods described in the literature (Thornton 1999a).

The VRM process has many phases, each using different techniques. Some techniques, such as QFD are useful early in the process, while others, such as KC flowdown are useful during assessment and mitigation phases. In order to create a tool that will aid, not hinder, the VRM process, the tool must be aligned with the process. Integrated tools and techniques will streamline the VRM process by reducing redundant work (Jay 1998). For example, increasing detail and modeling are needed as the process progresses. Early in the process, simple models must be quickly evaluated to narrow down concepts. Later, refined, complex models are used to predict specific behaviors and to optimize systems. Figure 2 shows how the breadth and complexity change with time. (Thornton 1999b)
No single practice can support the entire VRM process. Many tools and techniques are used currently. These need to be integrated into one tool or else linked together in useful way. Simply providing links to the information, without a way to use that information, is not sufficient (Thornton, Donnelly, Ertan 1999, Tata and Thornton 1999). New visualization techniques must be developed so that complex relationships and interaction can be displayed in an easily understood way. Current textual displays are useful for presenting lots of information at once, but not for making sense of that information. (Ardayfio 1998)

VRM activities are resource intensive and often are delayed because resources to support them are not available. These activities are also time-consuming and often do not focus only on the problem of variation. An appropriate VRM tool will reduce the amount of resources needed. Currently, risk is not identifies until late in the VRM process, and by then, the cost of mitigation increases. If tools are available to help identify and assess risk early in the VRM process, then costs can be reduced.
Documentation is a critical part of the VRM process (Jay 1998). The VRM tool provides a method of documenting VRM activities and acts as a repository for VRM information. Additionally, it can provide instruction in and references to a team’s VRM process. This instruction should define VRM goals, KC definitions, training procedures, and instructions on the process (Jay 1998).

2.4.1 Process Capability Database Research
Companies encounter many of the same obstacles implementing PCDBs as they do implementing VRM tools. Therefore, many of the lessons learned from the difficulties with PCDBs are directly transferable to VRM tool implementation. Many companies use PCDBs to monitor quality, but few use them effectively during design. Work by Tata and Thornton (Tata and Thornton 1999) details the obstacles preventing use of PCDBs during design and suggests ways to improve current practice.

The key problems identified are poor user interface design, lack of access to the PCDB, out-of-date data in the database, no interface to other tools such as PIM systems and VSA, sparsely populated databases, lack of training, and lack of compatibility between software systems across enterprises. The PCDBs will not be used if the incentives and processes are not integrated with the product development process. Even if the tool and process are linked, if users cannot get the right information from the tool quickly and in a format they can use, they will stop using it. The problems with PCDB use have caused several companies to withdraw support for them. The lessons from PCDB failures must be incorporated into VRM tools or they are subject to a similar fate.
2.4.2 Summary of Research-Based Requirements
Several of the papers by the VRM research group contain specific requirements for a VRM tool. The research-based requirements are shown in the Table 2.

<table>
<thead>
<tr>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support exchange of information within and between organization during all projects and all phases of the VRM process.</td>
</tr>
<tr>
<td>Support flowdown and traceability of KC information.</td>
</tr>
<tr>
<td>Support process capability information for use in all VRM processes, not just for monitoring capability.</td>
</tr>
<tr>
<td>Capture VRM information for use and reuse.</td>
</tr>
<tr>
<td>Provide information needed for measurement and inspection plans.</td>
</tr>
<tr>
<td>Provide instructions with clearly defined VRM goals and objective.</td>
</tr>
<tr>
<td>Provide training and supporting materials.</td>
</tr>
<tr>
<td>Ensure a consistent set of definitions.</td>
</tr>
<tr>
<td>Communicate what action is needed when a KC is identified to ensure understanding of the consequences resulting from a KC classification.</td>
</tr>
<tr>
<td>Provide methods to set customer requirement latitudes and flow customer requirements through the product.</td>
</tr>
<tr>
<td>Enable variation assessment early in the product development process.</td>
</tr>
<tr>
<td>Predict, record, and retrieve process capability and process capability uncertainty.</td>
</tr>
<tr>
<td>Quantify the impact of process capability uncertainty on yield and Cpk predictions.</td>
</tr>
<tr>
<td>Quantify the cost and benefit of mitigation strategies to select and optimally implement mitigation strategies.</td>
</tr>
<tr>
<td>Provide documentation methods that enable teams to develop a “system view” of variation.</td>
</tr>
<tr>
<td>Integrate the VRM tool with the VRM process and associated techniques.</td>
</tr>
<tr>
<td>Provide interoperability between tool and existing tools for VRM techniques.</td>
</tr>
<tr>
<td>Provide analytic and computational capabilities.</td>
</tr>
</tbody>
</table>

Table 2: Research-based Requirements for VRM Tool
(Tata and Thornton 1999; Jay 1998; Thornton, Donnelly, and Ertan 1999)

2.5 Interviews
Managers, systems engineers, people in manufacturing, and others at a partner company were interviewed about VRM practices and tools they have used. The interviewees were asked about the work they do and how it fits into the VRM process as they understand it. They were asked to give specific feedback on any tools they currently use and to describe
features of those tools they liked and did not like. If there were features they felt were missing from existing tools, they described those as well. Interviewees were passionate about the need for new tools to help them with their work. The interviews provided information about the requirements for a new VRM tool and about the VRM process as it is practiced at the company. It is important to understand the differences between documented and existing VRM practice when designing the tool.

2.5.1 Purpose of interviews

The interviewees were asked a series of questions about their role in the VRM process and the tools they currently use in this work. First, they were asked about their job description and about the day-to-day tasks their job entails. Next, they were asked about which VRM processes they are involved in and which techniques they use. Also, they were asked if the VRM process they use is the same as the one documented in company manuals.

The next questions concerned one of the current VRM tools used by a group in the company. Interviewees were asked how they use it, if at all, and what VRM techniques they use within the tool or using information from the tool. People inputting data into the tool were asked where the data comes from and what form it is in before being entered into the tool.

Next, there were questions about other VRM tools. Interviewees were asked what other tools they use for their VRM process, why they chose to use those tools, and how those tools fit with the VRM process. Finally, they were asked about their requirements for a
new VRM tool; what things do they like about their current tools, what things do they not like, and what features would they like to see in a new VRM tool which do not exist in current tools.

2.5.2 Interviewees
Eight people were interviewed. The interviewees were chosen because they each could offer a unique perspective. There were several systems engineers with different points of view and different amounts of exposure to existing tools, an administrator charged with entering data into the existing tool, a subsystem engineer, a manufacturing engineer, and a manufacturing process expert. One of the systems engineers also worked with manufacturing, using FMEA to trace problems that occur during production. Some people had used the current tool, while others should have used it but chose not to. Other people had never used the tool because its scope did not extend to their part of the VRM process. These people were chosen to provide input on how to expand the capabilities of a new VRM tool to include the whole range of VRM techniques across the VRM process. People were chosen who are concerned with different product levels; some focus on subsystem levels, others on system integration issues, and others on even bigger picture concerns. People in manufacturing were chosen because VRM information could be very helpful in fixing problems they encounter, but current systems do not flow down VRM information all the way to the manufacturing level.

The manufacturing engineer wanted statistical process capability (SPC) information, including distributions, included in the tool. Another engineer working at the manufacturing and design interface wanted flowdown techniques extended all the way
through manufacturing processes. A systems engineer working with Failure Modes and Effects Analysis (FMEA) also wanted manufacturing linked to the flowdown so that FMEAs could be used more easily to identify the root causes of problems. Another systems engineer wanted linkage with PIM systems and FMEA techniques. She also expressed the need for change control, a simple user interface, and reporting abilities. She said that no tool is useful if managers and high-level systems engineers can not get the reports they need. Yet another systems engineer wanted rules clearly inputted into the new tool. Since he works at the interface between systems, he sees the problems that occur when people have differing understandings of definitions and the VRM process. The last systems engineer, who chose not to use the company’s existing tool, also expressed the need for good instructions and documentation and communicated engineers’ desire for flexibility in their work. Unlike most of the interviewees, he preferred to use spreadsheets and did not want a wizard to guide him through the process. This demonstrates the need to a flexible tool. A subsystems engineer who works with groups in different geographical locations explained the need for flowdown, unit clarity, and the ability to use the tool for interactive management instead of just setting specifications. He wanted to be able to use the tool to aid communication with local and distant groups. Finally, someone tasked with using the company’s existing tool to enter data had much to say about user interface issues. His description of translating existing data in order to enter it in a different format showed just how important it is for the VRM tool to be integrated with the VRM process.
2.5.3 Summary of observations from interviews
Interviewees wanted a tool that will provide them with better access to information and better ways to communicate, without adding work. They wanted a tool for negotiating KC values at interfaces, understanding KC information, and communicating it to suppliers and other team members. People working with manufacturing wanted to be able to indicate that a tolerance was too tight. Systems engineers wanted to be able to look at an up-to-date state of the system in a way that made sense, not by looking at a listing of 150 KCs. Users want to be able to get information out of the tool in a way that makes sense to them. They wanted automated checking to see if values were entered correctly and if they concurred with existing data, and a way to view that information without looking at a wall sized chart. Engineers wanted flexibility in how they work. They wanted to be able to enter information in a way that made sense and view other information for comparison and to copy and paste. In a copier, for example, KCs for a cyan color cartridge may be related to those for magenta and yellow ones and a user may want to use many of the same values for the cyan cartridge KC. They wanted the tool to show them which other KCs were impacted if a change was made to the one they were working on.

A user of the current tool charged with entering existing data explained his frustration with translating from the documents used by engineers to the format used by the tool. He complained about having to move from the mouse to the keyboard and back and wished the interface would just use either the keyboard or the mouse at one time. Simple features such as spell checking, consistent tabbing (like in Excel®), comment fields, access to different fonts, super- and subscripts, Greek letters, and a consistent way of entering units
would have made his job much easier. He also wanted better descriptions for things so that it would be easier to understand what to enter and how it needed to be entered.

Managers wanted a tool that could track historical data and make predictions based on it. They wanted to be able to see the progress of the VRM process so far and use that to make predictions about future deadlines. Ideally, they wanted to be able to take information about progress of past programs and use that to predict the progress rate of current projects.

The interviewees all wanted a tool that would live up to the ideals described in the research and proposed concrete suggestions for how to achieve goals like a better user interface and integration with the VRM process. Their eagerness for the tool was apparent. They wanted it immediately, for the projects they were currently working on. A comprehensive VRM tool would provide immediate and significant benefit to organizations using VRM.

Some of the requirements from the interviews came from interpretations of what interviewees said. In some cases, they would give examples of specific cases where they were trying to express general needs and frustrations. Because the interviews were very open-ended and conversational in nature, the requirements came from the context of the story each interviewee told. Some came prepared with a list of requirements, in which cases specific needs show up in the requirements list. In other cases, interviewees wanted to express large, more pervasive problems they wanted the tool to solve. The following
sections use both the general and detailed information to provide requirements that reflect the desires of the interviewees.

2.6 Requirements

Prior research and interviews at companies identified specific requirements for a VRM tool. Other tools have been developed, but they have all had failings that limited their usefulness or rendered them useless to engineers in practice. The requirements for the VRM tool detailed here are based on company interviews and prior research and incorporate the best ideas in existing tools while improving on their flaws.

2.6.1 Consistency with Existing Practice

Current VRM tools do not reflect the VRM process as practiced in industry. They are generally glorified spreadsheet front-ends for a database of information. Engineers are told by their managers to enter data from their work into this database. Depending on what an engineer is working on, this often means translating data from a drawing or chart into the appropriate format for the database. This extra work does not directly benefit the engineer, as he still has to produce the original drawing, chart, or report from which he extracted the data. The information in the report or chart is not linked directly to the database. Very quickly, employees stop bothering to enter the information into the database. Other priorities and deadlines become more critical, and so the database is never populated completely. In other cases, early estimates are entered and then never updated. This sparse and unreliable database provides little help for people trying to get information from the database. In addition, the output is formatted in some format that
may be, although mostly likely is not, useful to the end user. Now the information must be translated again as it was when it was entered in the database.

The tool should look as similar as possible to the existing techniques. For example, if an engineer is used to using FMEAs, he will be able to view the data in that format. Similarly, when a user needs to get information from the tool, that information will be presented in the format the user needs. If a user needs a FAST diagram to insert in a report, he will choose the appropriate FAST diagram within the tool and then export it directly into the report. Different users will need to input different data in different ways. Instead of being presented with a large empty table, as many current tools do, the user will see a screen that looks like the part of the process he is working on at the time. The data fields on the screen will be linked directly to the database, so that as the user is going about his normal work process, he is simultaneously updating and populating the VRM database.

The front end user interface (UI) of the tool needs to be flexible to change. Different users will need to use different techniques when entering data, and the techniques will change over time. If the UI is not flexible, then the tool will become obsolete as soon as the next new technique is adopted, and the data in the database will need to be transferred to a new, expensive, tool which will require more hours of user training. The tool must be designed anticipating regular changes.
The tools must also provide a variety of input and output interfaces. In interviews with potential users, there was little consensus about what the best input and output formats should be. Each person wanted the format to match the techniques they used in their part of the VRM process. Manufacturing people thought there should be a focus on FMEA; engineers working with I-O-C charts thought those were most important. People want to use a tool that will help them do what they are already doing, not something that will give them even more work to do. A variety of input and output formats will allow many different users to store data in a common place without creating extra work.

In some cases, it may be more practical to link to existing software, especially for early versions of the tool. These linkages should not just be pointers to another document. They should provide a way to view data from other tools, and incorporate it into the VRM tool. Companies use many tools, and transitioning information from one tool to another can be time-consuming and costly. Providing a way to merge two tools together creates an easy, smooth transition. Examples of how to do this can be seen in databases that can read in spreadsheet formats and convert them to tables. Once these linkages are made, people who used to use separate, isolated software systems will be able to work together in a common environment. Communication will increase and problems will be resolved faster.

2.6.2 Platform Independence
Engineers and other users of the VRM tool all use computers in their work, but they do not use the same computers. People working on graphics-intensive design and modeling may use powerful workstations, while manufacturing engineers on the shop floor may use personal computers (PCs). The VRM tool must be able to run on any machine that a
potential user would use; it must be platform independent. By creating a platform independent tool that can run on any operating system (Windows, Unix, etc.) and any hardware, a company will not have to buy new computers just to be able to use the tool. Whatever existing computer network it is using will suffice. Platform independence is critical to widespread implementation of the VRM tool.

2.6.3 Information Control
Information control is the idea of identifying who has permission to view, change, add, and delete pieces of information in the database. There are great benefits to having so much information available to users, but if everyone has permission to do anything they want with all of the information, chaos will ensue. Instead, each user should have a set of things they are allowed to do with the tool, and the tool must know what these permissions are and prevent the user from performing actions that are not allowed.

Several techniques can be used to implement information control. Users must log into the system to verify their identity in some way. Once the system knows the identity of the user, it must grant the user only those privileges allowed by his permissions. Some users will only be able to update some types of information; perhaps only certain engineers are allowed to designate key characteristics, for example. Other pieces of information, such as specific physical part dimensions, will have owners who are the only people able to make changes.

The permissions and ownerships will add more information to the system. Not only will a user know what systems he needs to interface with, but also he can find out whom he
needs to talk with to coordinate that interface. Additionally, users will not have to worry that someone else will corrupt their data. Information control provides security for and from the users without limiting the capability of the tool.

2.6.4 Date Tracking
In additions to keeping track of users, the tool needs to keep track of the time history or events. In order to make predictions about how projects will progress in the future, managers needs historical data about how the project has been progressing to date. Date tracking keeps track of what has happened in the past and what should happen in the future.

One tool at a partner company provides graphics of the maturity of the VRM process over time. It examines how many parameters have been specified and with what degree of certainty. It identifies places where work has fallen behind a predetermined schedule. Interviews also reveal that users want to be informed about what steps they need to complete to stay on schedule. Does certain information need to be decided upon by a certain date? Is a specific graph needed for a report due on another date? Users want the system to be able to inform them of upcoming events, predict future trends based on historical data, and show the time history of specific programs.

Additionally, users mentioned a long-term goal of being able to use historical data from one project and use it to make predictions about a new project. This reuse of data could save lots of time and effort and allow for more efficient and accurate planning. Users
want to compare the current state of the project not just to past states of the same project, but to similar states in different projects.

2.6.5 System Level Views
Prior research by the VRM group has identified the importance of looking at data across the whole product, a system level view, in order to manage risk due to variation. These system level views are very powerful, but are not used today in industry because there are no good tools for viewing information on a system level. In many cases, the system level information is not even stored or understood. At a partner company, engineers each keep their own personal spreadsheets with the data about their part of the product. When a systems engineer wants to look at how the system containing these parts is doing, she has to go to each engineer and get a copy of his spreadsheet. She then has to put the information in all the spreadsheets together in a way that she can understand. By the time she does all of this, the data she is working with is a week out of date and she has to start all over. If all the information from all the lower level engineers was linked together, the systems engineer would be able to view instantaneous snapshots of the system.

Visualizing system level information, assuming it is all in one place, can be very difficult because of the amount of information involved. The full KC flowdown for a simple consumer product can take many pages, even when printed in small font. The user interface of the VRM tool must provide a way to help the user sort through the information available so that only the immediately relevant information is displayed.
Users also want a system that helps manage interfaces between systems and subsystems. Another tool at a partner company does this very well. Users must concur on parameters that cross system boundaries. Users must agree that they share this parameter and then agree on the value. This process is a negotiation, and the tool registers where in the process the two owners are. This system is an improvement on other systems that treat parameters on the boundary as the sole domain of one side or the boundary or another. If the parameter impacts both sides, there must be a negotiation. This additional information about the links between system elements provides an enriched system level view.

At every company we talked to, people expressed the need to understand VRM information on a product level. This is the key benefit and requirement for a VRM tool. Providing this ability in a way that is easily understood will be a huge leap forward for VRM tools and will accelerate the pace of the VRM process.

2.6.6 Data Consistency
With any database system, data integrity is crucial. Data must be entered in a consistent, correct form or the database is nearly useless. If units are entered as mm, millimeters, and cm, it becomes very difficult to sort and search the data. The problem is even worse if there are typos when the information is entered. The VRM tool must provide checks on the data as it is entered to ensure that data is valid before the database is updated. It must also perform consistency checks. Are the units for the nominal values and the tolerance both the same dimension (length, mass, etc.)? By performing as many of these kinds of checks as possible, the tool can prevent inconsistencies in the data that would make it hard to use.
2.6.7 User Interface
As shown by the problems with PCDBs and in interviews at a partner company, user interface (UI) issues are extremely important. User must have a working environment that is intuitive and easy to use. Users suggested incorporating features such as consistent screen layout, keyboard shortcuts, simple formats for searches, and graphical interfaces for visual representations. Users also expressed the need for choice. They want to be able to choose the most useful format for themselves when they need it. Since people could not agree on what the more useful formats would be, it makes sense to provide different options. All options must provide uncluttered screens that allow for quick access to the appropriate data. Ideally, these screens, and the tool as a whole, should be self-teaching. There should be clear instructions for use, glossaries, and menus that can guide users through the tool. With a good UI, users will see the tool as a help, not a hindrance, and will actually use the tool.

2.7 List of Requirements
The following list summarizes the requirements in the previous sections. Detailed needs, such as bringing up the right window at the correct time, were taken directly from the interviews. The rest of the requirements are based on either the prior research or interviews, although in many cases, the requirements were suggested by both sources.

User Interface
Intuitive
- Fits with current schedule
- Brings up the right window at the correct time
- Graphically based
- Windows matched to the process
Easy to use
  Minimum jumps between mouse and keyboard
  Every window must either be textual or graphical
Self-teaching
  The CP process documented
  Definitions and glossaries available
Provide necessary data without clutter
  Windows formatted to show necessary information
  Ability to choose the appropriate window
Simple commands/motions
  Any one interface allows mouse or keyboard input to use all functionality
  Keyboard shortcuts
Flexible format to allow user preference
  Users can choose a format as needed

Work with Other Software Tools
  Provide users on both ends with access to information they could not get easily before
    Link the information to facilitate communication and problem solving
Platform independence
  Allow for use on a variety of computer hardware and operating systems
  Incorporate or interoperate with analytic tools

Consistent with Process
  Presentation of information correlates with current phase of project and with job description
    User only sees the data relevant to his work
Guides user through process
  Knows what the process is
  Prompts user about important process items (due dates, reporting requirements, etc.)
Informs about deadlines
  System knows what the deadlines are for each user and for the project in general
  System knows what work must be done before each date and tells user when appropriate

Easy to input data
  Clear, consistent formatting
    A few basic window styles, that are all similar but customized for the application
  Able to view requested data as reference during data entry
    Able to view, and use other data, including copy/paste options
  Able to input in desired format
    Flexible entry formats matched to user need
    Graphical and textual interfaces as needed

Easy to get data in a desired format
  Formats must include any currently used formats
    (e.g. I-O-C charts, spreadsheets, graphical KC flowdown, etc.)
Easy to create a new format as reporting requirements change
Multiple output formats available
   Graphical and text based formats as needed for applications
   Multiple formats may be appropriate, and user should have a choice if so
Able to access and find information desired
   Simple formats to allow for easy searches
   Search and help functions to assist users
Historical data
   Provide historical data to show progress
   Reuse historical data as appropriate

Reconfigurable front end
Choice of input formats, default tailored to current phase
   Appropriate format brought up by default as determined by current phase
   in process and by job title
Ability to change front end (input/output formats) as needed
   Interface "language" to allow creation of new formats
Information control
   Only correct people can change data
   Owners of data clearly identified

Reusable back end
   Data structure must be robust to changes in information needs
   Able to be adapted to other industries

Help users with KCs
Provide guidelines
   Cue users about KC techniques as appropriate
   Provide reference for KC information
Provide linking and tracking capability
   Dynamic, graphical linking
   Visualization at different depths (zoom in/out)
Provide system view/flowdown
   Allow ways to view flowdown
   Assist users in creating the KC flowdown correctly
3 Information Architecture

Many different industries use the VRM process and the concept of KCs, yet between industries, different techniques predominate and different terminology is used. Even within an industry, there may be variation between companies, or even within divisions or projects at a single company (Ardayfio 1998, Thornton, Donnelly, Ertan 1999). Is it possible to address the VRM process in all these different places with a single VRM tool? This thesis identifies an information architecture that attempts to show that this is possible. This chapter outlines the design of this architecture and describes how its components work together to allow for this broad functionality. The details of the complete VRM tool design are discussed in the following chapter.

3.1 Problem Scope

The variation of the VRM process across and within industries and companies poses an interesting problem. Not only must the VRM process within a company or a group be identified clearly, but also the processes of different groups within a company, different companies within an industry, and different industries as a whole must be identified. It is important to understand the similarities and differences between the VRM process as it is practiced in these different organizations. By looking at the similarities and differences, one can develop a generalized process that is applicable to all the groups. This generalized process can then be used as the basis for a VRM tool that can be used by the widest possible user base. This section addresses the scope of such a tool.
Within a company there can be great variety in the detail of the VRM process. Different groups, or even different projects, will have different needs and will use different tools in their VRM process. At one of our partner companies, there are several tools used by different groups to track VRM data. Each group captures the data it considers most important. Some focus on capturing the linkages between parts of a system, while others track the progress made in defining the product but do not capture a system level view. Even when there is a consistent process defined across a company, that definition might be changed periodically to incorporate new advances in methodology and techniques. In other cases, the process definition may not be applied consistently between groups. Interviews at a partner company showed this to be a potentially serious, but very real obstacle to the proper implementation and adoption of a VRM process.

The VRM tool proposed in this thesis provides a way for engineers and managers to store, retrieve, and view product data in a manner consistent with their current VRM process, while simultaneously teaching them how to implement the process. It allows for intentional differences in the process across companies and groups by having a reconfigurable front end; the tool can be updated when practices and definitions change. A wizard guides the user through the VRM process, teaching the user the VRM process and ensuring it is correctly applied. The wizard provides hands-on training to reinforce the training companies provide about their VRM process and ensures that all employees implement the process consistently. This approach solves the problem of unintended differences in the VRM process across groups, but allows the flexibility needed to adapt to intentional differences between companies or groups. If the same tool can be used by
different divisions in a company, then employees moving from one group to another will not have to learn how to use a new tool. The wizard will teach the employee any new techniques they did not use before, but the common elements and the general user interface will remain consistent and provide a familiar work environment.

Additionally, the VRM process must be reflected in the user interface. The user interface must be as flexible as the rest of the tool. Some users may prefer to manipulate data using graphical techniques, while others may prefer to use raw data. Interviews in industry reveal that employees will not use a tool with a poor user interface, yet they could not agree on what a good user interface would be. This difference in preferred user interface is not discussed in corporate descriptions of the VRM process, but is crucial to the implementation of a useable tool. The tool must be flexible enough to adapt to the demands of different users. The problem of user interface design must be addressed in design of the tool, taking into account the existing body of knowledge about user interfaces.

These three sources of differences in the VRM process, industry-, company-, and user-based, provide additional challenges for the design of the VRM tool. To keep the scope of this thesis manageable, only some of the variation questions are addressed.

3.2 Software Techniques

Two software techniques are used to implement the requirements of the VRM tool. The Model-View-Controller (MVC) framework, is a way of separating the information in the
computer, the display shown to the user, and the mechanism by which changes are
effected in the data model. Software layering is the separation of parts of a software
system through carefully defined interfaces. Each of these concepts, and the way in
which it enhances the VRM tool, are discussed below.

3.2.1 The Model-View-Controller Model
The Model-View-Controller (MVC) model was developed for the Smalltalk-80
programming environment (Krasner and Pope 1988). Since then, the MVC model has
been used in many applications of object-oriented programming (Pomberger and
Blaschek 1996). The MVC is one of many design patterns used in object-oriented software
development. Design patterns are “solutions to recurring problems in software
development.” (Govoni 1999)

The MVC model supports interactive development of graphical user interfaces (GUIs).
The model divides applications into three parts, the model, the view, and the controller.
The data structures that represent the screen display are the model of reality. The display
shown to the user is the view. It deals with everything graphical, requesting data from the
model and displaying it. The controller is the interface between the models, views, and
input devices. It processes mouse, keyboard, and system input. Manipulating the display
using the controller effects changes in the data model. (Pomberger and Blaschek 1996.
Krasner and Pope 1988) A simple diagram of the MVC model is shown in Figure 3.
The view and controller always occur in pairs, since a controller is needed to manipulate the display. In light of this, modern references often refer to the MVC pattern simply as the Model/View pattern. Because the controller and view are bound so closely, they are referred to together as the view.

In the VRM tool, the model is all the information about the product variation, the VRM process, and the people using the tool. The view is the representation of that data, the KC flowdown, the I-O-C chart, etc. The controller allows users to manipulate that view. For example, if the user is creating a KC flowdown using a graphical interface, then by dragging the mouse between two KCs, he can connect them in the flowdown. The controller realizes that the user has used the mouse to connect the two KCs and sends a command to update the model. The model gets updated to reflect the new connection between KCs and notifies the view that there is updated data. The view then gathers the data from the model and refreshes the display for the user to represent this updated model.
The MVC model is the basis for the interaction of the different components of the VRM tool. It isolates the functional elements and makes it easier for the designer to understand and modify each element, without understanding or impacting the other elements (Krasner and Pope 1988). The tool also takes advantage of software layering, described below. Together, these techniques provide the encapsulation of the VRM tool components and describe the interactions between them.

3.2.2 Software Layering
The VRM tool uses layered software techniques to enhance flexibility. In layered software, the smallest, most fundamental details are at the lowest level. At the next level, are the basic functions that use the details at the lowest level. Another level higher will use details from the level below it. For example, if basic arithmetic functions were at one level, A, a higher level, B, could allow for more complicated equations with grouping. Some function in B could break the equation into a sequence of basic arithmetic and then use the appropriate sequence of functions from level A to solve the equation.

Software can have as many layers as appropriate.

To manage the layers, specific interfaces must be defined between layers. These interfaces describe what kind of functions can be called and what parameters they require. Once these interfaces are defined, it is easy to add new elements at a layer. This is very useful when new functionality is needed after the original design has been implemented. The data within each layer is encapsulated. It can be accessed through prescribed interfaces, but is otherwise protected from the outside. In layering, each encapsulated layer will communicate with only those above and below it, not with all the other layers. Modules within a layer may communicate with any other modules in that
layer as necessary. An example of how communication between encapsulated works can be seen in Figure 4.

![Figure 4: Encapsulated Data](image)

Software layering seemed like a natural approach to use for the VRM tool since the information architecture of product development data lends itself to this kind of organization. Layering provides convenient abstraction barriers that allow programmers to focus on the specific detail they are implementing. Defined interfaces allow different implementations of modules to replace old ones when needed, and new modules can be added using these existing interfaces. A layered approach allows each component of the software to perform only simple tasks, relying on how these simple pieces are put together for complexity, instead of building one large complex module. This collection of simple functions is far easier to comprehend, more adaptable when changes are need (and they always are), and far superior for debugging, reuse, and sharing code with new
programmers (Brooks 1995). Software layering is the natural approach for a VRM tool because of its ease of use as well as its similarity to the organization of VRM data.

The VRM tool uses six layers in its design. Five of the layers are vertically connected and only have interfaces with the layers above and below them. The sixth layer provides system administrators access to the system and interfaces with all five other layers. The high-level design is shown in Figure 5. The five vertical elements flow naturally from the VRM process data. The lowest level layer, the data structure, provides the fundamental building blocks for the rest of the structure. All the data used by the rest of the layers is stored somewhere in the data structure. The next layer, basic functions, defines the simple interactions between these data structure elements and the user; these could include adding, deleting, and modifying elements, and looking at the relationships between them. The process elements layer provides ways for the user to use the basic functions. The process elements are higher level modules that use combinations of the basic functions to create a representation of the basic data in a form useful to the user such as a spreadsheet or flowdown view. The process description captures the entire VRM process as practiced by a company or group, which process elements are used by whom and in what order. The user interface provides the front-end encapsulation of all the other layers. It provides a windowing system and ways of navigating through the tool. Combined with administrator access, these six layers capture the data resulting from the VRM process as well as data about the process in a form intuitive to the user. Each software layer is discussed below. A detailed discussion on the data structure layer follows in the second half of this chapter, and the remaining layers are addressed in Chapter 4.
3.2.3 Software Layers

**Data Structure.** The data structure level contains the data from and about the product development process. This data is the raw material used by the VRM tool, and the data structure it is housed in is analogous to a well designed warehouse. The data is stored in a relational database as a set of objects. Each object contains information about its type, attributes, and connections to other objects. Detailed documentation of the data structure level can be found in later sections.

**Basic Functions.** The basic function level contains functions that manipulate the data structure database. Functions such as creating a new data object or changing an existing one are common to all process elements. These functions are analogous to cutting and forming in a manufacturing process. The database can only be changed using the functions in this layer. This provides a way to insure that only valid changes are made to
the data. These simple functions are the lowest level interaction between the process and the data.

**Process elements.** These are the tools and techniques used to organize and manipulate the data. The process elements layer is the tool chest of the VRM tool, containing a multitude of ways to use and change the data in the data structure warehouse. They are the components of the VRM process. Examples include creating an I-O-C chart, flowing down KC’s, and linking KC’s. As new techniques are developed, they can be added as new elements. The elements use the basic functions in the lower level to get information to and from the database.

**Process description.** The process structure describes the product development process by stating which process elements must happen in which order and at what time. It is the process plan for the engineers and managers involved in the VRM process, telling them what tasks need to be performed, when they need to be completed, and which tools are needed to do the task. The order of steps in the process can be changed as needed just by editing the description. Each step in the process calls one of the process elements. Steps may be recursive if needed. The process description for a particular project or group is entered by a manager who determines the schedule and methods appropriate for that project. The system uses this information to guide people through the product development process. It gives users reminders of deadlines, tell them which data they need to develop for future dates, and remind them of unmet deadlines. This information
allows the VRM tool to have a wizard to teach users the VRM process as they work through it.

**User interface.** The top-level control provides a framework for the rest of the VRM tool. This is the source of windowing, menus, and other resources that allow the user to navigate the software. New windows or menus can be created if needed, although it is expected that this will not need to happen in most cases. While this front end can be adapted as needed for any one company, keeping a consistent front allows users to continue using the system without having to relearn it often. Even when modules in the lower layers change, the user interface can remain the same. This consistent interface provides a stable work environment for users, even if their process changes, resulting in less time spent learning a new system and more time spent doing value-added work.

 Additionally, there needs to be a way for a system administrator or programmer to access the system. He is responsible for implementing any changes to the tool and all VRM tool maintenance duties. By adding new elements and configuring existing ones, the administrator is able to customize the system for a particular company or group. General users should not be able to do this or the tool will wind up a hodge-podge of uncoordinated bits. The administrator will also be responsible for backing up the data and other system level tasks required for the tool’s upkeep.
3.3 Data Structure Elements

Companies use many different structures to organize and represent their VRM data. Some structures, like product architecture, are used by all companies. Other structures reflect the way information is used within a product organization. Through reviewing company documents detailing product development processes, and through conversations with industry partners, a rich data structure was developed for use with the VRM tool.

This formally defined data structure allows for a common back end for wide scale VRM software systems. It is the model in the MVC architecture. The research provides an understanding of what the data is and how it can be used. Understanding the connections provides new views of the product and the product development process. The VRM tool combines the best practices used in industry. The connection between these practices empowers the user to identify and resolve problems faster and to create a better product for less. The data structure common to these techniques provides the backbone for a very powerful tool.

The data in the data structure layer is broken into a structure with several elements. The data structure elements, product architecture, KC architecture, organization structure, budgets, and customer requirements and environmental variables, contain the basic data for all product development activities. The elements relate to each other in specific ways; each element in the KC architecture has an owner in the owner list, each product architecture element may be related to an environmental variable, etc.
3.3.1 Product Architecture

The product architecture is the “scheme by which the functional elements of a product are arranged into physical chunks and by which the chunks interact.” (Ulrich and Eppinger 1995) It describes the physical elements of the final product (e.g. features, parts, and subassemblies) and how those elements fit together. This is the architecture most engineers are familiar with because it can be seen by looking at how a product is assembled and put together according to an assembly drawing. The product architecture must be included in the VRM tool since users are so familiar with it and expect it to be part of their work. Assembly drawings provide this view, and it is the way most work is structured. Systems, subsystems, and parts each have someone, or an area of people, responsible for them. The product architecture extends all the way down to manufacturing and the bill of materials.
The product architecture is a tree structure. Each level of the tree is a different type of product element. The highest level element is the final product, next down is the assembly level, followed by subassembly, part, and feature. It may be possible to have multiple levels of subassemblies. Each lower level element is the child of the higher level element of which it is a part. Children may have multiple parents. For example, several subassemblies (the parents) may contain identical fasteners or brackets (the children).

A simplified airplane product architecture is shown in Figure 7. The product, the airplane, has four children, the wing, engine, tail, and fuselage. The tail, in turn, has two children, the tail support and the rudder. This example could be expanded to include every rivet, skin panel, and hydraulic tube in the aircraft.

![Simplified Airplane Product Architecture](Ardayfio_1998)

Each element of the product architecture tree contains the feature name, feature type, owner, and drawing number where the feature is defined. The product architecture represents physical data. Therefore, any data associated with physical objects is stored here. The drawings are linked since they also represent the physical product.
One table in the database contains all of the elements. Another table contains the links between the children and the parents. Each child must have at least one parent, or else the element will not be linked with the rest of the product and will not appear in the system level view. The highest level element, the product, has no parent; it is designated specially as the top node. Tables provide a convenient method for storing sets of data in a consistent format. The two tables allow changes to be made to the linkages between elements without modifying the elements and vice versa.

3.3.2 Key Characteristic Architecture

![Key Characteristic Flowdown](image)

**Figure 8: Key Characteristic Flowdown**
The key characteristic (KC) architecture shows how variation, and the cost and risk associated with it, flow through the product in a KC flowdown. The KC flowdown shows how the different levels of KCs are related to each other. It describes both the amount of variation in a feature and the sensitivity an upstream part has to that variation. KC flowdowns are critical to implementing the VRM process. The flowdown enables a “system view” of the product. With this view, engineers can optimize designs on a product or system level instead of locally, and during production, problems can be traced easily to their root causes. A product level KC, perhaps a customer requirement or a government regulation, will generate related system level KCs, which in turn will generate subsystem level KCs. A KC can flow up or down to multiple parent and child elements. In Figure 9, KC $H$ impacts both $E$ and $F$, and KCs $G$ and $H$ both contribute to KC $E$. The KC flowdown can be continued through various subsystems and parts to the feature level. The cost and benefit of mitigating the variation can be traced similarly through the tree structure.

Figure 9: KC Flowdown with Sensitivities and Multiple Parents and Children (Thornton 1999a)
Each KC element is linked to at least one feature in the product architecture. In some cases, it may be related to multiple parts if the KC describes the relationship between two or more features. For example, the gap in a car door may be a KC. The gap is determined by the fit between the door edge and the doorframe. In this way, the KC architecture is linked to the product architecture.

The KC architecture, like the product architecture, is stored in two tables. KC architecture elements contain the name, KC type, nominal and latitude values (with units) and the ID number of the associated feature. The KC type can be product, system, subsystem, part, or feature level. There is also information about the owner, the last time the element information was updated, and process capability and cost information. Similarly to the product architecture, one table contains information about each element, and the other contains information about the connection between elements and attributes about that relationship including sensitivity, uncertainty, and percent contribution.

3.3.3 People
In a corporate environment, there is always someone responsible for each part of a project. The ownership of any feature must be known for rapid communication and problem solving to occur. For example, an engineer working on a climate control system chooses a new, more efficient motor. She realizes that the KC flowdown shows the output voltage requirements, a KC, will not be met by this new motor, and after tracing the flow further, realizes that this output voltage is critical to another system. She can now reconsider her design choices or identify the people responsible for the impacted systems and negotiate the change with them. (Ardayfio 1998)
Each feature in other tables is tied to an owner, linking the product data to the people in the organization. Organizational data is stored in a table of potential owners. Owners are people who have responsibility for or authority over some part of the data in the VRM tool. Users can find person responsible for features they are working with, and through the owner data structure, can find the contact information for the owner.

Because implementing an org chart in the VRM tool would be a duplication of effort and provide an extra database to keep current, one is not included in the people data structure, although it does list the group they are affiliated with. On the rare occasions where someone needs to know someone’s position in the organization, he can look that up in the same place he currently would. Each employee listing includes the person’s name, location, email address, phone number, and mail stop. SQL queries allow users to find the system elements each person is responsible for.

3.3.4 Budgets

![Diagram of Budgets]

**Figure 10: An Example of a Budget**
Budgets are system outputs that can be aggregated and must be managed. They capture places where several elements taken together produce a common output that must be controlled. Examples of budgets include temperature, audible noise, and reliability. In each case, a design may call for a maximum or minimum allowable value for the total output of a system. The budget manager must allocate the budget for each subsystem and negotiate with the subsystem owners when conflicts arise. The budget is a summation of all its contributing elements across the system or product, so one element can not go out of bounds without impacting the others.

Budgets are not KCs. KCs look at the risk and cost of variation and how that contributes to high level requirements. Budgets consider the total values of certain outputs without considering variation or how the values impact each other. They are used in parallel with the KC structure, but do not duplicate it.

Budgets are linked to product and KC architectures. For each budget, a new table is created in the database, and a new field is added to the KC and product architecture element entries. Not all KC and product elements will be impacted by each budget. In this case, the budget field for that element will indicate that there is no impact by that budget. A table of all elements impacted by a budget can be generated through SQL (Structured Query Language) queries.

3.3.5 Customer Requirements and Environmental Factors
Some customer requirements and environmental factors (CREFs) directly and significantly impact many elements of a product. Environmental factors can include
maximum and minimum operating values for atmospheric pressure, humidity, and air quality. Customer requirements can include copier paper size and fuel composition. These are important inputs to many systems, but are externally determined and therefore not variable. For each new product, these factors must be determined early in the product development process. CREFs function similarly to the KC architecture. They are inputs to many elements and identified separately for convenience. They will be entered into the VRM tool, and then that information will be used to guide engineers as the work on the rest of the design. As users input data for a new part of the design, they will be prompted to state whether each factor might be an input to that element of the design. If it is, the factor will be linked to that element as a parent element.

3.3.6 Company Specific Elements
Companies will have other special designation that only they use. These cases can be dealt with by adding columns to the appropriate tables. If a company specifies specific people as KC experts, there can be a field added to the people structure to identify whether a person is a KC expert. If certain KCs are called something else because they need special treatment, there can be a field added in the KC structure to distinguish those KCs. During discussions about the VRM tool at partner companies, people seemed very interested in having the tool reflect their terminology and practices. Allowing for company-specific flags enables this requirement. The VRM tool is designed to handle changes like these, and even the addition of new kinds of table if needed.
3.4 Summary

The VRM tool uses the MVC model and software layering to implement the data structure and techniques of the VRM process. The data structure provides the foundations for the rest of the tool. The other layers act on the data in the data structure using the basic functions. The process elements implement the VRM techniques and the process description organizes them into a coherent VRM process. The user interface and administrative controls provide ways from users and administrators to interact with the tool.

The data structure elements reflect the information in the VRM process. The KC and product architectures show the functional and physical relationships between product elements. The people structure keeps track of users and their contact information. Budgets help manage outputs with constraints on the total output, but not on specific outputs. CREFs designate factors that are likely to impact or be inputs to many elements of a product. The basic data structure, along with any company specific additions, provides the model in the MVC design and the common back end for the VRM tool.
4 VRM Tool Design

The VRM tool manages VRM data, implements VRM techniques, and guides teams through the VRM process. The tool design is guided by the requirements described in Chapter 2. Its design allows each implementation to be consistent with current and local VRM practices. A system level view permits teams to optimize for and understand the system or product and to see how different components impact the final results. Information is controlled so that the correct users can modify the right data and so that all data is screened before it is entered into the database to assure data consistency. The tool keeps track of what changes are made when and is able to provide historical data for reuse, observation of progress, and prediction of completion dates. The user interface incorporates the requirements described in Chapter 2 so that people will prefer using the VRM tool to the methods they use now. Finally, the tool is platform independent, so that companies can use in whatever computing environment they prefer.

The tool design uses the layering concept discussed in section 3.2.3. Each software layer builds on the information in the layer below it. The lowest layer, the data structure, contains the raw information from and about the VRM process in an organized manner. The next layer contains the basic functions that operate on this data. These functions are then built up into process elements that are the actual VRM techniques teams use. The process description describes when, how, and by whom these techniques are used so that the tool can guide people through the process and inform them of deadlines and reporting requirements. The user interface provides the framework for the tool and a consistent working environment for users. Administrative access is provided for maintenance of the
tool. This chapter discusses each of these layers, excluding the data structure, showing how each higher level builds on the ones below it. Additionally, the end of the chapter discusses the software needed to implement the tool.

4.1 Basic Functions
The basic functions are the interface between the VRM database and the rest of the VRM tool software. They provide a way to view, add, remove, and change elements in the data structure. The only changes to the database can be made using the basic functions; the database can not be accessed directly. This protects the database from accidental corruption. Since the basic functions modify the data structure, they are part of the MVC controller.

4.1.1 Database Connectivity
First, before being able to make changes to the database, the tool must establish a connection to the database. Through this channel, the software of the VRM tool can communicate with the VRM database. Depending on the number of users, the amount they are using the tool, and the type of database, the tool may open a new connection each time it needs to access the database or it may keep a connection open as long as the software is running. The database may or may not permit simultaneous access from multiple users and may limit the number of connections at a time or over a period of time. Depending on the organization, a more or less robust tool may be needed. Software engineers should choose the database wisely when implementing a production version of the VRM tool to find one that matches their needs. If many simultaneous connections will be needed, then the database must be chosen so that it, and the supported bridge
interface, can support them. Otherwise, users may not be able to connect when needed or they may be locked out of the system. If fewer connections are needed, then a different, smaller, less expensive database can be chosen. Once the connection to the database is established, the other basic functions can be used to interact with it.

4.1.2 Adding, Changing, Deleting, and Viewing Elements
The basic functions provide a way to view, add, change, and delete elements as necessary for each data structure element. Some fields in an element, such as the name and owner, are necessary to being able to use that element. Other fields, such as percent contribution, are not necessary for entering an element into the data structure, although they may be needed to perform tasks. Finally, other fields are entered from computer generated data. For example, the KC ID is created by the database when a new KC is created, and the VRM tool software generates a new last update timestamp each time an element is updated in the database. Each of these kinds of fields, necessary, optional, and automated, allows and requires different viewing, adding, changing, and deleting methods.

When an element is created, all necessary fields must be specified. The necessary fields contain the minimum data required to insert the element into the data structure. For example, when a KC element is created, the user must specify the name, type, and associated part number. When a product element is entered, the user must specify the name, type, and owner. If a field is not specified, then the software must return an error. Users should be able to view and change necessary fields as they wish. However, they should not be able to delete values from necessary fields unless the entire element entry is
deleted. If an element does not have all its necessary fields, then it can not be connected correctly to the rest of the tables in the VRM data structure.

Optional fields can have values added, changed, deleted, and viewed whenever needed. Since the value is only needed to perform higher level operations, it can be omitted and still leave the data structure intact. For example, architecture elements may not have an associated drawing number when they are created, so it would be entered later in the VRM process. Nominal and latitude values of KCs are also optional fields. They are necessary for quantitative analyses and for determining appropriate manufacturing processes, but not for identifying that there is a KC to be monitored.

The automated fields should never be entered or changed by the user. For example, a KC ID should be created by the database and then left untouched unless the element is deleted. The last update timestamp should be updated only when an element is created or another field is updated successfully. It would be incorrect to update just the timestamp if there have not been any changes to the rest of the data. All automated features are added, changed, and removed automatically by the software or the database. Users may view the automated data whenever they need, but they can not alter it.

Some data types may require new tables to be created or fields to be added to existing tables. In this case, the adding functions not only create a new field, but also populate that field for each entry in the table, as needed. For example, when a budget is created, a new entry is added to the list of budgets, and a new field for that budget is added to the
product architecture elements table. Initially, the values in new field are set to zero until the budget manager determines budget requirements for each contributor.

Users can manipulate individual fields of an entry when appropriate, but they are required to enter data for a sufficient set of fields when creating new entries. These elements of the basic functions only provide a way to manipulate and view the database. Higher level functions provide a way to do something with that data.

4.1.3 Data Caching
Users need to have fresh data to work with. If they are working with data that is out-of-date and are not aware of revisions to that data, then their work is wasted. If the tool constantly downloads new information from the database, the user will always have fresh data, but the database may be overwhelmed. Gathering data infrequently risks the user having stale information. Data caching examines when the data a user sees was last updated from the database. If the last update was longer than a certain time ago, the tool gets fresh data from the database. Depending on the number of users, the amount of time they use the tool, robustness of the database to high traffic, the tool refreshes the data at different rates, usually a few minutes between updates. Software engineers should factor data caching needs and abilities into their database choices when designing production versions of the VRM tool.

4.1.4 Data Verification
Data must be correct, as well as fresh. The VRM tool examines data before it is actually entered into the database to ensure it is of the correct form. If a field is supposed to take integer values, then attempts to enter non-integer values produce an error. Company
partners commented on the problems they encountered with a tool that did not auto-check data. Because users could enter data incorrectly, searches of the data would not produce correct results and the database was no longer useful. If a value must be the same as a different value somewhere else, then the tool checks that it is. For example when creating a KC linkage between a parent and a child, the values for the child and parent IDs must be the same as ones already in the KC element table. Data verification preserves the integrity of the database, without which, the database quickly becomes useless and unmanageable.

In some cases, data verification means restricting the values a field can have to a short list of accepted values. For example, many data elements contain the type of the element. For KC elements, these types are restricted to product, system, subsystem, assembly, subassembly, part, and feature KCs. When there are only a few choices for the values a field can take, then the tool ensures that the user knows what those choices are and is only allowed to enter values from that list. If users are allowed to enter any terms they wish, then they will come up with synonymous terms and the database will not be able to find elements of the same type. If synonyms are desired, then all the synonymous terms must be entered into the tool so that when it performs a search for a particular type, it can also search for all synonymous types.

The code for the basic functions implements the data verification. Each method used to enter data first checks that it is in the correct form (e.g. integer, text, etc.) before allowing it to be entered. For type support, the user interface only provides the ability to enter one
of the valid types. This is implemented by only allowing the user to choose from a specific menu or by not accepting entries that are not from the specified list.

The basic functions provide a buffer and an interface between the VRM database, and the tool and the user. They allow for a connection between the tool and the database, a way to access and manipulate appropriate values in the database, and a way to ensure that the values entered are valid entries. Data verification maintains the integrity of the data in the database, as specified in the requirements. The basic functions do not provide a way to manipulate the VRM data and extract additional information or insights from it. Those features are provided by the process elements.

4.2 Process Elements

The process elements organize, view, and manipulate the data in the data structure layer using the elements in the basic function layer. The process elements are the VRM techniques, the pieces of the VRM process as it is practiced in industry. They are both views and controllers in the MVC architecture. The views are the graphical representations of the data and the controllers are the functions that act (through the basic functions) on the underlying data structure. Process elements can include simple tasks such as sorting and searching elements of the data structure, data type specific tasks such as summing all the elements of a budget, and more complex VRM techniques such as the KC flowdown or I-O-C charts.
The specific process elements used by a company will depend on the techniques they already use in their VRM process. Each company's VRM process is different because of such things as regulations, the nature of the business a company is in, and its relationship with its suppliers (Jay 1999). Not all techniques are appropriate for all companies. System administrators can choose to include the appropriate set of process elements depending on the needs of the organization. If the needs change later, new elements can be added and old ones removed without losing any of the data in the database. The process elements only provide a way to view the data usefully; they do not store the data.

A process element can, in combination with the basic functions, provide the basis for a different process element. Simpler elements, such as searches, may be common to several VRM techniques. The simple searches are not included in the basic functions layer, however, because they arrange the data in the database into a different, more useful, form. Similarly, data type specific elements produce new data based on the initial data, so they, too, are considered process elements.

4.2.1 Search and Sorts
The vast amount of VRM information in the database necessitates searches and sort. Users simply cannot make sense of all the data at once, so they must be presented only with the portion of data that is relevant to their current needs. Simple searches can produce lists of elements that match the given criteria (e.g., the lists of all KCs owned by Henry Smith, all the parents of a given subsystem KC, the list of elements that belong to the heat budget.) More complex searches can look recursively and find the whole family tree of a particular KC, all the parents and children. This is useful for looking at the KC
flowdown. By understanding where variation flows to and from, assessment and mitigation of KCs improve.

During mitigation, sorting can show a prioritized list of places where mitigation will be useful. Sorting can speed other searches. When looking at historical data, it is often useful to see how many KCs are at different points of maturity. If the KCs are sorted by maturity, determining this takes very little time.

The following elements are the core searches and sorts needed to implement the VRM tool.

**Search for owner** finds all the elements owned by a particular person. This information, along with information about scheduling should allow the tool to prompt the user about elements they are responsible for that are behind schedule.

**Search by type** in the KC and product architectures finds all the elements at a particular level. This can then be used to help visualize the flowdown. For example, a user might want to see the flowdown only to the system level. The VRM tool would then search for all the elements at and above the system level and display those.

**Search for parent** and **search for child** enable users to move up and down the product and KC architectures. These searches should allow users to specify the element they are interested in and then view how that element interacts with other parts of the system. Recursive versions of these searches should find all the parents and all the children of an
element, showing all the elements in the flowdown that impact or are impacted by an element.

Search by budget finds all the elements that contribute to a particular budget. This is useful for generating reports for the budget manager who must keep track of all budget contributors.

Several sorts are needed, to organize the presentation of large amounts of data.

Sort by update shows where data has been updated most frequently.

Sort by maturity shows which KCs are most mature and which still need more work. For example, some KCs may only have names and linkages to their parents and children, while others may have nominal and latitude values set. This feature exists in some existing tools and is used by managers to see if progress is on schedule. Users also need to be able to perform basic sorts on any other fields.

When working with data at a partner company, we realized that we wanted to organize the data in new ways to point out particular areas. For example, we wanted to view all the KCs that were associated with a particular system listed by their subsystem. In other cases, users may want to view all the KCs by drawing number so that they can conveniently see all the KCs they need to watch for on specific drawings. Users need to be able to perform basic sorts on any fields they deem necessary.
These searches and sorts provide ways to manage the vast amounts of data in the VRM tool so that it can be presented to users in the form they need. They enable flowdown visualization and support flexible viewing techniques. There are a variety of commonly used search and sort algorithms that can be used to optimize performance in production versions of the tool. Any text on algorithms should provide complete coverage of the benefits of each. Some will be better when searching or sorting over certain kinds of data; some searches require previously sorted; other searches only work if the data set is small enough. Common sense and an understanding of the different searches should be applied during implementation to ensure quick and efficient operation.

4.2.2 Data Type Specific Elements
Data type specific elements perform actions on specific parts of the VRM data structure. Budget specific elements should include adding a new budget to the data structure by adding an entry to the budget table and adding a new field to the product architecture elements, summing the contributions to a given budget, and comparing that sum against the maximum value allowed for that budget. For CREFs, users need to be able to add and remove factors from the CREF table, and simultaneously, add or remove the links to the CREF in the KC flowdown. When a user wants to set a CREF as an input to his system, the VRM tool must automatically link the CREF into the flowdown in the correct place. The data type specific elements often will be unique and not part of established VRM techniques. Still, they provide valuable information to the people who work with this data, and must be included in the VRM tool.
4.2.3 VRM Techniques
The VRM tool implements the VRM techniques as modules in the process element layer.

The techniques use other process elements to present data in a format that is immediately useful to the user. A variety of techniques should be implemented to align with all parts of the VRM process. This section presents examples of two techniques, I-O-C charts and KC flowdown visualization.

I-O-C Charts
I-O-C (input-output-constraint) charts list the input, output, and constraining parameters for a module (system, subsystem, etc.) of a product. These are sometimes referred to as P-charts (Phadke 1989). Linking all the I-O-C charts creates a functional flowdown. Engineers use I-O-C charts to manage KCs and to figure out which other systems they need to communicate with to be able to negotiate and modify their inputs and outputs.

The I-O-C chart is really three tables put together. The input and output tables look nearly identical except the input table tells where the input comes from and the output table tells where the output goes to. The constraint table tells what the constraint is, but does not have information about where it goes to or from since it is considered a fixed requirement.

The I-O-C process element is fairly simple. It contains three tables and an identification bar across the top, as shown in Figure 11. Underneath, is an optional viewing window. The window should allow users to reference data from other charts, and can be hidden when not needed. The three tables list the inputs, outputs, and constraints. The identification bar tells which module the I-O-C chart describes and who owns the
module. The input table, on the left, lists all the input parameters to the system. Each entry includes the module from which the parameter flows, and the number, name, and nominal value (with units) of the parameter. The output table, on the right, contains the same information about the output parameters, but the first column lists the module to which the parameter flows. The center column, the constraints, lists the number, name, and nominal value (with units) of the constraints. The add, remove, and sort buttons at the top of the optional viewing window allow users to add, remove, and sort elements in the optional window. This provides the ability to reference other information easily, as specified in the requirements.

![Figure 11: I-O-C Chart](image-url)
Users enter information into the I-O-C chart, as they would in a spreadsheet application. If they want to modify an entry, they simply move the cursor to it and modify the value. When users first open the I-O-C chart for a module, the VRM tool automatically enters data from the KC flowdown into the appropriate input, output, and constraint tables. When users delete parameters they are prompted to check that they meant to remove them. Any time parameters are added, removed, or modified, the KC tables are modified to reflect the changes. If users need to reference additional information to enter their data, they can copy and paste from the optional window, but they can not modify any of the information in it, as it is just a reference view.

If users want to look up new additional information, they click on the Add button at the top of the optional window. This pops up a screen where users can type in the name or ID number of the parameter they want, or search through parameters for the appropriate one. If the user highlights a parameter and then clicks delete, he will be prompted with a screen asking him if he really wants to delete this parameter from the optional window. Users can also highlight and delete multiple parameters at once. Users can also sort the parameters in the optional window by number, name, or module by clicking on the sort button and choosing one of the options. The optional window allows users to view any additional information they want in the least obtrusive way possible.

While the I-O-C chart can be used to create a KC flowdown, it does not present the flowdown in a way that is intuitive for the user. I-O-C charts are best for viewing and modifying the KC element fields and for seeing all the parameters for a particular
module. For a real understanding of the flowdown, graphical representations, like the KC flowdown, are best.

**KC Flowdown**

KC flowdown visualization provides a system level view of the VRM data. It shows how variation flows on a system level. The flowdown is a graphical technique used for creating and viewing the KC architecture. This section details how the flowdown process element should be implemented. The user interface section discusses the “wizard” interface that guides people through creating or modifying a flowdown if they are unfamiliar with the process. The discussion begins with an overview of the screen layout and then goes into more detail about each required feature.

![Figure 12: Basic Flowdown View](image)

Figure 12: Basic Flowdown View
The screen shown in Figure 12, has three main components; a menu bar at the top, a graphical view in the main screen, and an optional details panel to the side. The menu bar contains all the commands needed to manipulate the flowdown view. The main menu commands also have keyboard shortcuts so experienced users can work more quickly. If the user does not want to see details about the flowdown elements, he can hide the details panel and have more space for the graphical view. The details panel contains information about the KC and allows users to update the KC information, assuming they have permission to do so. All users are permitted to view the KC flowdown, even if they cannot change it.

User may add, delete, modify, connect and disconnect KC elements or entities. To create a new flowdown, the user creates a new entity. He enters the basic information for the entity in the details panel and a square appears on the screen to indicate the new entity. By clicking the right mouse button on the square, or by using the menu bar, the user can create children or parents of the KC element. Each new element is created by entering information into the details panel. Users fill in data for each field they want, but must enter all the necessary fields for the element to be inserted into the flow. Elements can be deleted, but a warning occurs before a parent element is deleted. If a parent element is deleted, then the children may not be linked into the flowdown. Similarly, KC elements can be connected together, but a warning message is given if disconnecting two elements will result in elements not linked to the flowdown.
The fields in the details panel, shown in Figure 13, provide the information about the highlighted KC flowdown element. As users scroll the cursor over each element, the element information appears in the details panel. Additionally, the name and type of the KC appears in a pop up window if the cursor lingers over an element. This allows users to get the basic flowdown information without needing to have the details panel visible. The name, type, part, nominal, latitude, and Cpk fields can each be filled in, but only the name, type, and part number must be entered. The user picks the type from a scroll down list using either the mouse or keyboard shortcuts, therefore ensuring that the type is a valid choice. Users may enter the associated part by typing it or by browsing for the correct choice in a pop-up screen. If an invalid part is entered, then the user is prompted to enter a different part. When the user wants to enter nominal, latitude, and Cpk values, he may. Again, the units are chosen through a scroll down list. The list has two parts, the
first menu determines the dimension type (e.g. length, voltage, etc.) and the second gives options for that type (e.g. mm, in, μm, etc.) This way, units are entered consistently, and the tool does not have to know that $10^{-3}$ m is the same unit as mm.

Users can also enter data about the connections between KCs elements. This information shows up in the details panel if the link is highlighted. Each link can contain information about uncertainty, percent contribution, and sensitivity. This information can be used later with quantitative techniques during the assessment phase of the VRM process.

Once the KC information is entered into the tool, the user will want to view the KC flowdown in an understandable manner. The KC flowdown process element provides ways for users to manipulate the view of the flowdown so that they only have to view the information they want. Users can enter display parameters to choose which information to display, and then use other tools to indicate how to present that information on the screen.

The display parameters window, shown in Figure 14, allows users to select the categories of information they want to view. Other information will be shown in a light gray, as shown in Figure 15, so that the framework of the flowdown is still understandable, but users can easily identify the elements of interest. Users may chose to view KCs by owner, maturity, Cpk value, or to see all KCs. If companies have designations for “special” KCs, an option to view just those KCs can be added to the display parameters window. The example in Figure 15 displays the Show Family option.
Figure 14: Sample Display Parameters Window

- Special Characteristics
- Owned by
- Maturity: High, Medium, Low
- Cpk greater than
- Show all data

Figure 15: Parameter Choice, Show Family
Users may want to focus on just a part of the KC flowdown at one time. Several utilities allow them to change the display of the flowdown to show just that information. Using the display menu options, users can show all the KC elements or just the family of a particular KC (all grandparents, parents, children, grandchildren, etc.) They can leave the flowdown on the screen but shrink parts of it, as shown below, so that only part of the flowdown is shown at the normal size. A sample screen is shown in Figure 16. Users can shrink the parents, children, or draw a box on the screen and shrink all elements inside that box. Users can also decide to hide parts of the flowdown in a similar manner to shrinking. This eliminates those elements from the view, but stubs from the remaining elements indicate that connected portions on the view are hidden currently.

![Figure 16: Example of Shrinking Flowdown Areas](image)

88
A few more utilities help navigate the display on the screen. Users may pan up and down and left to right across the flowdown if it does not fit onto the screen at once. If the KCs start to appear as a jumble of boxes, the alignment utility will arrange them on the screen so that siblings are aligned neatly.

**Conclusion**
The VRM tool can implement any of the VRM techniques. There should be a range of technique available so that people working in all parts of the process will have tools available that will address their concerns. The tool must be matched to the process or it will not be used. There should be graphically based tools and text based tool. Some people greatly prefer spreadsheets to more graphical presentations of data, while others find lists of data utterly useless. If the same data can be presented in multiple ways, more people are likely to find a way of viewing the data that is intuitive to them. The input and output formats for the tool must match existing reporting requirements or the company must change their reporting requirements. If someone can produce a beautiful graph with the tool, but can not produce the graph he needs for his meeting, the beautiful graph is useless to him. The outputs should be available in different file formats so that people can incorporate their results into a number of other programs.

4.2.4 Reconfigurability
The modularity of the process elements allows the VRM tool to provide a reconfigurable front end. When a new technique is needed, only the software that implements that technique must be written. The infrastructure, the ways to interact with the database and the generic user interface elements, already exists. This reduces the amount of time needed to add new capability to the tool. Additionally, because the process elements are
modular, adding a new element will not change anything about the existing elements or how they interact with each other and the data structure.

Each time the system is configured for a particular company or group, the software engineer can pick the elements that are needed and just include them in the setup. If other elements are needed later, they can be added then as plug-ins, the way users can install additional elements of the Microsoft® Office® suite after they have the basic installation. This provides a lean tool that adapts to the needs of the company or group using it.

New process elements are written using the software’s API (Application Program Interface). This interface defines the allowable ways to interact with each process element, basic function, user interface, and administrative tools. When new elements are created, they must work with and use the API. Their interface specifications must be added to the existing API so that future programmers will know how to interact with the new elements. A well-defined API will allow for ease in programming new process elements.

4.3 Process Description

The process description defines how the process elements, the VRM techniques, fit together to form the VRM process. Certain techniques must be performed before others. Certain information is required before other information can be determined. Some of the time, processes must be repeated once new information is developed, forming an iterative loop. The tools stores information about who needs to perform each task and the dates
and deadlines associated with that work. By recording the order of activities, showing which come first, which come next, and which are concurrent or iterative, the tool can inform users about where in the process they are and what tasks must be completed to reach a deadline. This information, in conjunction with references to how the process elements should be performed, guides users through the VRM process and teaches them by having them actually go through the process.

The process description can be modified for each project, or even during a project if plans and schedules change. Users with the appropriate permissions, probably managers, can modify the information in the process description. The process description is yet another way the tool is flexible to the specific, local needs of an organization. The quickly and easily modifiable process description allows for big changes with only a little work.

Each process element is broken into several tasks. For example, creating, developing, and confirming the values in an I-O-C chart are all different tasks. When the I-O-C chart is created, some of the inputs of outputs may not be known, and the chart is just roughed out. During development, each input, output, and constraint is entered into the chart so that there are no missing pieces. Finally, the values in the chart must be confirmed to be the correct ones. Each of these tasks may occur at different times, even though they are all part of one process element.

The information about each task is contained in a task element. When organized together these elements describe the entire process. Each task element contains a task ID, the task
name, a list of people responsible for the task, the scheduled and actual start times and stop times for the task, and the pointer to the file for that task. A comparison between the actual and scheduled timing of a task can be used in reporting to show whether the process is on schedule and to identify where things are falling behind. This supports the reporting requirements expressed my managers. The method for organizing the task elements is discussed below.

The Design Structure Matrix (DSM) shows the interaction and dependencies of the different task elements. Each task element should be assigned to a row and the corresponding column. A mark indicates that the task element in that column must be completed before the task element in that row. By reading across a row, one can see all the task elements that must be completed in order to perform the task element corresponding to that row. Reading down a column shows where the information from that column’s task element is used. If the elements are arranged in the order they are performed, then the DSM is sequenced or partitioned. Once the DSM is partitioned, it is easy to identify sequential, parallel, and coupled tasks. (Ulrich and Eppinger 1995) A simplified DSM for the Kodak Cheetah project is shown in Figure 17. It shows examples of sequential, parallel, and coupled tasks.

Each task element is an element in the DSM, one of the rows. The task elements contain information about each task, and the DSM organizes the tasks so that the VRM tool can prompt people about what they need to do when. The “wizard”, described in the user interface section, is responsible for this.
The DSM is stored as a matrix coordinated with a list of task elements. Each position in the matrix records whether there is a check in the corresponding location in the DSM grid. The task list contains all the tasks in order, so the tool knows which task corresponds to which matrix row or column.

The author studied the VRM process at a partner company to show how the DSM model would work with their process. This company’s process is very well documented and understood by the employees. In the interviews to determine the requirements for the VRM tool, interviewees were asked to describe their place in the VRM process. Their answers revealed that they understood not only their part of the process, but also how their work fit in with the work of others. They described which VRM techniques they used for each step of the process. A DSM was created based on the information from the interviews and the company documentation. At the start of a new project, a manager would create a similar, project-specific DSM that would be used in the VRM tool.
The process description, in conjunction with the modularity of the process elements, provides the reconfigurability described in the hypothesis and the requirements section. The process element layer design allows administrators to change which elements are included in the tool. The process description allows them to change the order of those elements.

4.4 User Interface

The user interface (UI) provides another way to make the tool easy for people to use and learn to use. It is clear from the requirements that a well-designed user interface can make the difference between a tool people will use and one they will not. The UI provides a way to organize and present the components of the tool in a way users can understand. It also provides general utilities such as save, print, and help functions. "Wizards," guides to using the tool, provide a convenient way for new users to learn about the tool and produce results quickly. An example is shown later in section 4.4.2. The process elements make the VRM tool useful, but the user interface makes the tool something people want to use for their work.

4.4.1 Basic User Interface Elements

The user interface provides the windowing, menu, keyboard shortcut capabilities for the VRM tool. Computer users are familiar with using windowing systems with menu bars in their work. Most common software applications use this technology. The VRM tool UI provides the windowing framework, but the content inside specific windows is determined by the particular VRM technique being used. This is similar to using a web
browser that pops up a PDF reader inside the window when you want to view a PDF file. You can still use the basic menus of the web browser, but you can also use PDF specific tools that pop up in a menu bar at the top of the screen. The windowing environment encapsulates the process elements and provides a common operating environment for the user, reducing learning time for each process element.

The windowing environment must allow users to open multiple windows at once, similar to the way many applications allow users to view multiple files at once. Users will want to compare files, see what work was done previously on a different project, and view the same data in multiple formats so they can make more sense of it. Users also need to copy and paste between the different windows. If a new program is based on a similar program from the past, users will want to copy information from the old program instead of starting from scratch.

Users need both menu-driven methods and keyboard shortcuts for performing routine tasks like copying and pasting. New users need to be able to search through menus to find a way to do what they want. Experienced users (power users) will be frustrated going through the menus each time, and will prefer keyboard shortcuts (e.g. Control-P to print.) (Brooks 1995)

The menus in the windowing system should be clear and intuitive. Certain menu options, such as file, edit, and help, have become de facto standards for top level menus. Many
groups are researching how to create the best user interfaces. Their results should be considered when determining the final layout of the user interface and the menu options.

In addition to easily understood menus and keyboard shortcuts, power users need an ergonomic tool. If a user needs to keep switching between using the keyboard and using a mouse or other pointing implement, he will get muscle fatigue and will enter data more slowly. When possible, screens should allow users to use either the mouse or the keyboard to perform tasks, but minimize the simultaneous use of both. Other ergonomic factors include choosing easily typed keystroke combinations for the most common shortcuts (Brooks 1995).

The basic UI elements provide a consistent, intuitive, and easily learned working environment for users. Thoughtfully designed windows, menus, and keyboard/mouse combinations will produce a comfortable working environment.

4.4.2 Additional Capabilities
Several added capabilities help users interact with the tool and simultaneously teach the VRM process. Incorporating local terminology, help functions, and wizards for common tasks will help create a self-teaching tool that is consistent with current, local VRM practices.

Using local terminology avoids confusion with company specific definitions and provides a tool that matches the company’s VRM practice and terminology. If a company uses the term Critical Parameter (CP) instead of Key Characteristic (KC), then reference to
KCs would confuse users while references to CPs would allow them to refer to supporting training and documentation and understand better what they needed to do. Local terminology should not be hard-coded into the process elements because it may change without the process elements changing. Local terminology is a critical piece in implementing a tool consistent with current, local VRM practices.

Help functions and searches provide users with guidance and information about the VRM process and techniques. There must be a searchable glossary of terms, including company specific definitions, and more detailed explanations of the VRM process, including examples. Much of this information already resides in company specific VRM documentation. A searchable glossary will allow users who are unsure of the specific term they need to find what they are looking for. The detailed explanations of the process will provide extra reference for users so they do not have to go through three binders of training materials to find the information they need to do their work. These help functions should complement other help the tool provides, such as explanations about how to use the tool and “wizards” to guide users through the process.

The help functions are implemented using a help structure. For each help topic, the corresponding help structure contains the name of the help topic, the text about that topic, a list of related topics. When appropriate, pointers to an example and/or a link to start a “wizard” are included. When users search for a help topic, the help function will display the text from the appropriate help structure, as well as the related topics and links to an
example or wizard. This way, users can find help even if they are not completely sure of what topic to ask for.

The “wizards” will guide the user through performing most routine tasks. When a user wants to create a flowdown, it will ask him for a starting element and prompt him to enter the necessary information for that element.

Please enter the name or number of the first element of the flowdown:

Name

Number

EXIT  SEARCH

Figure 18: Starting Flowdown Wizard Prompt

Next, it will ask him to enter any parents or children of that element and provide him with dialogue boxes for entering that information.
Once the basic flowdown is created, the tool will explain how to modify the flowdown by adding additional elements or changing information about the existing elements. The “wizards” allow new users to generate work quickly and teach users about the basic capabilities for the VRM tool.

The wizard uses information from the process description and the help files to guide the user through the process. When a user needs to perform a specific task, the wizard will display the appropriate help file to explain how to do that step. If the user wants to understand the next step, or get more information about how this step fits with the rest of the process, he can look at related help topics that are suggested at the end of the current help file.

When a user logs in, the wizard checks the task elements for any tasks assigned to that user. It then sorts out the tasks that have been started but not completed and any new tasks scheduled to start. The wizard then tells the user which task he should work on. If
there are multiple current tasks, the wizard will ask the user which step he would like to start with first. The wizard then starts the process element associated with the current task, and if this is the beginning of the task, the wizard displays a tool wizard, as described above. Otherwise, the wizard pops up the final screen of the tool wizard reminding the user of how to continue working with the particular process element.

Besides helping with each step, the wizard also provides an overview of the process. When a user logs in, it will give him a message letting him know which tasks he needs to do, and what the deadlines are. It will let him know if another user has entered information he needs into the system by reviewing the DSM. If all upstream DSM task elements have been completed, then the user will be informed that he has all the information needed to complete the task. Otherwise, he’ll be told which tasks are still incomplete. The DSM only indicates which tasks need to be completed before others can be completed; it does not show when tasks can start. Some tasks can not be started until prior ones are complete, but others can start earlier and then be completed once the information from previous tasks is available. The wizard will also look at the process description tell him about upcoming tasks, so he can see how the current work will fit into future efforts. In this way, the wizard guides the user through the process, as well as the individual steps of that process.

The VRM tool user interface is one of the most important parts of the tool. As shown in the requirements section, a poor user interface can doom an otherwise promising tool.
Intuitive features like the windowing and menu systems, combined with extensive help and “wizard” functions will create a self-teaching, easy to use working environment.

4.5 Administrative Controls

The administrative controls provide a backdoor into the VRM tool. Administrators can add new users, add new process elements, and provide other services such as backing up the data in the database. Administrators can access parts of the system otherwise off limit to the normal user. They can access the database directly and modify elements in any other software layer. Obviously, they should avoid direct access to the database whenever possible and just use the basic functions as needed. However, should something malfunction, they can step in and fix it.

The layered nature of the software will make it easier for administrators to test, debug, and solve any problems that might occur. They can test elements at each layer until they come to the layer that does not work. This will narrow down the problem to a more manageable scope. The VRM tool should be robust to many problems, but no piece of software is perfect and it is prudent to design in the ability to fix things when they go wrong.

4.6 Software Issues

The VRM tool is composed of the database of VRM information and the reconfigurable front-end that provides an interface to the data. The database can be any commercially available database package that provides for interaction with a software language,
typically through SQL. The reconfigurable front-end must be implemented in a software language that can interface with the database. The database can run on any platform that will be used as a server in an organization's infrastructure. Some operating systems will require a “bridge” between the database and the operating system. The bridge is a piece of software that provides a way to access the database from the specific programming language. There are many commercially available bridges, depending on the database and programming language chosen.

A platform independent language such as Java™ should be used so that the same version of the tool can run on any operating system. Java™ is an object-oriented operating language. This means that the data structures and basic functions used in the tool are easy to create and control with Java™. Current versions of Java™ also include Swing™ that provides a convenient way to create and manage user interfaces and internationalization support. Internationalization allows the same software code to be used in different countries and still conform to the local style guidelines for things such as date and currency formatting. Java™ is a developing language. Developers constantly produce new toolkits making it easier to perform tasks with Java™. For example, toolkits can add graphical capability, optimization methods, or cryptographic security tools. Each programming language provides advantages over another and new programming languages are always being created. Developers should choose the language that easily implements the most features of the VRM tool, while still permitting implementation of all desired functions.
4.7 Conclusions

The VRM tool provides a way for users to view, store, and manipulate VRM data in a manner consistent with their current VRM practices. The layered design and administrative access allows administrators to modify and add elements to the tool with a minimum of time and hassle. VRM data is protected through data verification. Process elements incorporate current VRM techniques and provide new techniques for looking at a system view of VRM information. The user interface combines basic UI elements with extended capabilities to provide an intuitive, self-teaching front end. The tool, based on specifications from users and research in the field of variation risk management, creates a way for teams to increase their communication and documentation skills while improving their implementation of the VRM process.
5 Conclusions and Suggestions for Future Work

The VRM process provides companies with a way to increase quality while decreasing costs and reducing design cycle times. The VRM tool supports the VRM process by enhancing communication, providing documentation about the VRM process and a central repository for VRM data, and by permitting system level views of that data.

The author hypothesizes that there is a common, underlying data structure across companies that captures information from and about the VRM process. This data structure was developed based on the prior research, analysis of the VRM process at a partner company, interviews with people implementing the VRM process and discussions with attendees at the annual VRM/KC symposium. Data from a partner company was entered into the data structure. This test showed that the data structure accurately captures the VRM information.

The hypothesis also proposes a flexible, reconfigurable tool to align with and support the VRM process. Interviews and research show the need for a reconfigurable VRM tool. Users need and want different ways to view and use VRM data. It is not feasible for companies to purchase and learn a new tool for each part of the VRM process. There is too much variation between parts of a company and between companies to allow companies to pick an inflexible tool. A reconfigurable tool allows companies to use one tool for many different parts of their VRM processes, and connects all the techniques to one common database, so information is captured and shared more easily.
The thesis proposes a design for a reconfigurable VRM tool. The tool’s basic functions were implemented and shown to work with the common back-end data structure. The design of the rest of the tool builds on this foundation and shows how to create a complete VRM tool.

This work describes the structure and framework of the VRM tool. More work needs to be done to expand on this foundation. The following sections discuss areas of focus for future research and development of the VRM tool.

5.1 More Testing of the Data Structure

The data structure behind the VRM tool was designed using the extensive research about VRM practices across industries and by reviewing company documentation about VRM techniques. This information was compared with interviews with industry partners to create an accurate view of actual VRM practices. Some company data was used and compared against the data structure, but a complete test was unable to be completed. The preliminary tests showed the data structure to be flexible and able to capture the VRM information from the partner company. Before a final version of the tool is created, a full test of the data structure should be performed. Existing data from several projects should be inputted into and outputted from the system using the various process elements. If the techniques permit this use of the data, then the tool will be shown to match the existing data.
5.2 Explore Usability Across Industries

Because the research foundation for the VRM tool includes surveys of many different companies and industries, it is expected that the tool design will support implementation across industries. Case studies should be performed to identify any areas where the tool does not capture the difference in VRM processes in different industries. The tool must be configured for several companies in different industries and then used to see if the tool aligns well with each company’s VRM process. Ideally, the VRM tool would be common to all industries and provide a standard platform for communication and documentation. If suppliers and customers were all using the same software to capture their work, information transfer would be greatly simplified.

5.3 Support for Electrical and Software Designs

Modern products are no longer predominantly mechanical or electrical; they combine mechanical, electrical, and software systems into one integrated unit. Industry partners asked how the VRM tool would capture the requirements and flowdown in electrical and software systems, since those are just as important to the success of a product as the hardware systems. A review of company literature shows that companies usually have parallel, but separate, processes for developing their hardware, software, and electrical systems. An in depth study of these practices, similar to the work done in developing the VRM process, could show how VRM techniques apply to these areas and suggest ways that the VRM tool could support the electrical and software development processes.
5.4 User Interfaces
User interface design is an active field of research within computer science. Computer scientists and others are trying to develop methods for presenting information and gathering information from people. While the user interface of the VRM tool is critical to its success or failure, this thesis does not develop strong recommendations for a user interface implementation. Instead, the focus of the thesis work has been to show how a VRM tool can support the VRM process. Different user interfaces are needed for different types of end users; designers, managers, and manufacturing specialists all require different ways of interacting with the VRM data. A reconfigurable front-end provides the flexibility needed to accommodate each of these different users.

A final implementation of the VRM tool must include a carefully designed user interface that takes advantage of this extensive field of research. Sample user interfaces should be tested with potential users to identify the best implementation, and the final version should incorporate the best features of each prototype version. As with any consumer or software product, alpha and beta testing will lead to the best result.

5.5 Visualization Techniques for System Level Views
Visualization techniques are the focus of ongoing research activities. Presenting VRM information to the user in a comprehensible way is very difficult because of its complexity. New techniques, including 3D representations of the data, may provide a way to present the data in an understandable way. The KC flowdown is a powerful technique, but the cross interactions between KC elements make it hard to view a system
of any size in its entirety. Even in small print, the KC flowdown for a relatively simple product can span multiple pages. Until visualization techniques are developed, KC flowdown techniques will be difficult to implement.

5.6 Closing Remarks

The VRM tool uses a layered software approach and a common underlying data structure to create a reconfigurable front-end interface. The reconfigurable front end allows the tool to be matched to the current VRM practices at an organization, and to be changed if those practices change. The common data structure back end stores VRM data regardless of the format used to input or output that data so that it can be viewed at a later time using any desired techniques. By allowing companies to implement the VRM process, sooner, better, and throughout more of their product development process, companies can expect to improve quality and lower cost.

Industry understands the importance of reducing costs, improving quality, and lowering design cycle times in today’s economy. The VRM process provides a way for companies to do all these things. The VRM tool provides a way to implement the VRM process consistently and correctly. The software provides a way to manage and present VRM information.

Companies are using computers for tasks that previously occurred only on paper. Some computerization attempts have been highly successful, while others have failed. The VRM tool attempts to learn from these lessons and provide a way to manage information,
encourage communication, support documentation, and teach the VRM process simultaneously.
References


