AN ABSTRACT OF THE DISSERTATION OF

Kerry R. Poppa for the degree of Doctor of Philosophy in Mechanical Engineering presented on August 22, 2001

Title: Theory and Application of Vector Space Similarity Measures in Computer Assisted Conceptual Design

Abstract approved:

Robert B. Stone Irem Y. Tumer

A number of computational tools now exist to aid in developing conceptual solutions based on a functional description of a design problem. A key limitation of these tools is the way results are organized for presentation to the user. In general, results are an undifferentiated mass of potential solutions. Analysis using a novel concept clustering tool shows concept generator output represents permutations of a set of a few solution archetypes. This provides an initial solution to organizing and presenting the results. More efficient solutions are sought by adopting a generate-evaluate-guide framework from the computational design synthesis literature. Specifically, the concept generation approach is altered so that each generated solution maximizes the variety it adds to the set of solutions. To achieve this, suitable similarity measures must first be developed.

Current techniques for similarity assessment in the design literature tend to be ad hoc and highly specialized to particular tasks. Prior work from the field of information retrieval is applied and extended to create a generalized approach to similarity assessment for vector space design data. These techniques are validated against an existing design by analogy methodology. A new tool for locating functional analogies within a database of existing products is developed as a result.

Improved similarity measures are combined with the proposed computational synthesis framework from literature to modify an existing concept generation tool. The resulting tool efficiently locates the few novel solutions in the set of possible results, and is a key step in the continued evolution of this class of computational design tools.

Theory and Application of Vector Space Similarity Measures in Computer Assisted Conceptual Design

by

Kerry R. Poppa

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosopy

Presented August 22, 2011
Commencement June 2012

Doctor of Philosophy dissertation of Kerry R. Poppa

Presented on August 22, 2011.

APPROVED:

Co-Major Professor, representing Mechanical Engineering

Co-Major Professor, representing Mechanical Engineering

Head of the School of Mechanical, Industrial, and Manufacturing Engineering

Dean of the Graduate School

Kerry R. Poppa, Author

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my co-advisors Dr. Rob Stone and Dr. Irem Tumer for their guidance, support, and advice during my graduate studies. I am indebted to my committee for their time, patience, and thoughtful feedback. This work has been influenced by collaborations with kind and helpful colleagues from a number of other institutions. Their feedback and insights have molded my thinking over the years, and I am in their debt.

While working on this dissertation I was supported by funds from the National Science Foundation (CMMI-0742677, IIS-0841379, and CMMI-0927745), Oregon State University Mechanical Engineering Fellowships, and the G&E Lundstrom Scholarship. Any opinions, findings and conclusions or recommendations expressed are mine and do not reflect the views of any of these funding sources.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF TABLES

LIST OF TABLES (Continued)

DEDICATION

I was blessed to have been born into a family that gave me a love of mechanical things, and then tolerated by unending questions, and occasionally poor assumptions, about how they worked. This dissertation is for them, and for A and E who joined me on this misadventure.

1 Introduction and Background

1.1 Introduction

Automated concept generators, a class of computational design synthesis tools aimed at aiding conceptualization in early design, have shown great promise as aids to designers. However, the present generation of these tools requires the designer to manually guide the exploration of the solution space, severely limiting their utility. Present theory on computational design synthesis posits a three-step loop of generation, evaluation, and guidance. At present, the guidance step is off loaded to the user; this must be rectified before these tools can move forward to wider adoption and use. This dissertation applies vector space query matching techniques from information retrieval to solve the problem of guidance in automated concept generation. This dissertation closes the loop for automated concept generators by providing evaluation of generated solutions and guidance in the generation of new solutions. These tools are aimed at conceptual design so novelty is a primary metric for evaluation concepts. What is needed is a method to automatically assess the similarity of generated concepts.

1.1.1 Roadmap

The chapter begins by discussing opportunities for enhancing automated concept generators. A key finding is that there is a need for widely applicable techniques for assessing the similarity of engineering artifacts across a variety of dimensions. A series of research questions that will be answered by this dissertation is outlined. A brief synopsis of the organization of the remainder of the dissertation is given. Finally background on key concepts and tools is presented.

1.1.2 Contributions of This Chapter

1. Opportunities to contribute to the state of the art in automated concept generation literature are identified
2. The need for universal approaches to design artifact similarity measurement is established
3. The literature related to methods and tools relevant to this dissertation is surveyed.

1.2 Motivation and Research Questions

Many engineering design activities require the practitioner to assess qualitative or quantitative similarity between components, systems, and phenomena, often across domains and disciplines. A task as simple as catalogue design requires the designer to evaluate how closely an existing component matches perceived requirements, while activities like design by analogy and concept generation using morphological analysis require assessing similarity at increasing levels of abstraction. Current techniques tend to be ad hoc and highly specialized to particular tasks.

Deficiencies of current similarity measures tend to go unnoticed because the human mind is adept at classification tasks, and can often correct for errors or inconsistencies. Despite this advantage, measures are needed to move this evaluation process towards a computational setting. Evolution may have adapted the human mind to solve these problems, but we are not immune to significant cognitive bias. The volume of data now readily available compounds the problem. Given the many viable information sources available to us, many problems may be intractable for the un-aided human mind. If the computer is used as an aid, then it must be recognized that it shares none of our innate ability to sort and categorize, but brings an objective lens to the evaluation process. Thus, measures of similarity are needed which are insensitive to our biases, able to handle meaningfully large sets of data, and efficiently computable.

These issues are not unique to engineering design. Modern information retrieval techniques locate documents most relevant to a given query using matrix techniques that produce quantitative measures of similarity. This dissertation addresses the need for quantitative similarity measures in engineering design by answering three key questions.

1. Can automated information retrieval techniques be adapted to provide similarity measures for engineering design methods?
2. How does the performance of these adapted techniques compare to existing methods?
3. If the proposed technique is suitable for similarity assessments between existing products, can it be further adapted to evaluate automatically generated design concepts and guide automated concept generation?

Information retrieval posits a variety of techniques for query matching that may be adapted as measures of similarity for engineering design. In general these techniques begin by constructing a vector space model of the data for comparison. While the appropriate data for comparison is largely determined by the problem at hand, a contribution of this dissertation is to demonstrate appropriate vector space representations of product function and morphology. An existing repository of engineered products will be used as the primary product data source for the proposed work. Once a vector space model has been constructed a matrix approximation of it is calculated.

While there are many applications for a quantitative similarity measure of engineered artifacts, answering the first two research questions requires a more narrow scope. Though a number of useful engineering design activities could be used to validate the application of information retrieval techniques, this dissertation focuses on design by analogy for two reasons. First, there is recent exploration of design by analogy in the literature suggesting that this is an area of interest to the research community where there may still be room for contributions. Second, the literature contains quantitative similarity measures for design by analogy that can serve as benchmarks to assess the proposed work.

The third research question turns the focus back to computational design tools. This dissertation initially explores automatically clustering concept generator results after generation. Rather than sorting a set of generated concepts, the techniques developed are used to map each concept into a reduced vector space representing relevant product knowledge as it's generated. This mapping guides a concept generation algorithm that seeks to maximize the distance in the vector space between the previously generated concepts and the next. This allows the most novel solutions to be generated within the first few iterations of the concept generation algorithm, eliminating the need for extensive post processing of large sets of generated concepts.

1.3 Organization of this Dissertation

This dissertation is organized into six chapters. Each begins with an introduction, a roadmap of topics covered in the chapter and a summary of contributions made. Each chapter ends with a conclusion briefly summarizing key results. This section briefly describes the outline of this work to help the reader jump directly to relevant chapters.

Chapter 1: Chapter one discusses the motivation of this work and lays out a basic overview of the research presented. Key background information relevant to the entire dissertation is presented here as well. Where possible background specific to the content of each chapter is presented just prior to new work

Chapter 2: Chapter two presents a first attempt at addressing the problems inherent in the current generation of automated concept generators. Rather than attempt to modify concept generation algorithms, it sorts the results selecting the most useful or relevant to present to the designer. Achieving this requires developing a scheme for estimating concept parameters from a design repository and employing techniques from exploratory data analysis.

Chapter 3: Chapter 3 reviews current work in function based analogical design and applies current practice to compare inter-product similarity among systems in a design repository. This provides the necessary benchmark to tackle the first two main questions of this dissertation.

Chapter 4: Chapter four identifies and applies techniques from information retrieval that can provide a universal approach to vector space similarity measurement in engineering design. These tools are applied to the functional analogy problem introduced in Chapter 3.

The new method of similarity measurement is shown to compare favorably to the method of Chapter 3.

Chapter 5: Chapter five modifies the concept generation algorithm and applies the techniques tested in chapter 4 to guide the generation of a few novel solutions from the set. Some problems from Chapter 2 are revisited with the new concept generation algorithm.

Chapter 6: Chapter 6 summarizes the conclusions of this dissertation. Key contributions to the literature are highlighted. Future directions for automated concept generation, computational design synthesis, and analogical design are discussed.

1.4 Background

The following section reviews relevant background information on a number of concepts relevant to this work as a whole. Other background a literature review sections are spread throughout the text to keep important prior work close at hand as new techniques are developed. The material in the following sections is presented here, at the beginning of text, because it has some bearing an all the work that follows.

1.4.1 Structured Design Methods

Significant effort has been devoted to studying and algorithmically describing a systematic process for engineering design. Suh has developed axiomatic design which provides a framework for transformations between needs, functionality, physical embodiment, and manufacturing process, as well as set of fundamental axioms by which the design activity can be judged [1, 2].

Altshuller advanced TRIZ, a design methodology concerned with overcoming technical contradictions and emphasizing novel solutions [3], though often confused with its central algorithm TRIZ attempts to present a complete design methodology. The proposed work, most directly follows from the systematic design methods originating in Europe in the last century[4, 5], and which have been expanded on in a variety of design texts [6-9]. These approaches advocate solutions driven by customer or societal needs, functional problem decomposition and development of complete solutions by aggregating partial solutions to product sub functions.

While there is still debate on its individual steps and boundaries, the four broad activities of engineering design, shown in Figure 1.1, can be identified: Preliminary design consists of exploration to understand a problem or identify an opportunity; conceptual design consists of exploring the solution space to propose conceptual solutions consisting of arrangements of components that can be evaluated; embodiment design consists of defining physical parameters of the selected concepts to initiate prototype development; and detailed design consists of engineering analysis, manufacturing details, and engineering drawings to support the final manufacture of the product or system.. The proposed work is primarily concerned with enhancing and augmenting the designer during the conceptual stage of design.

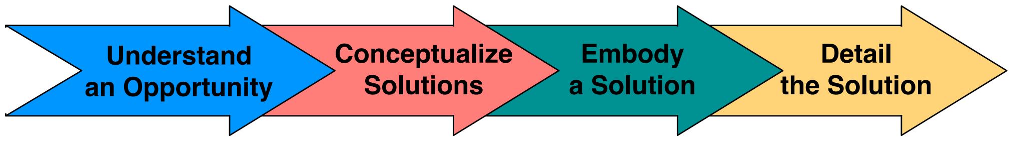
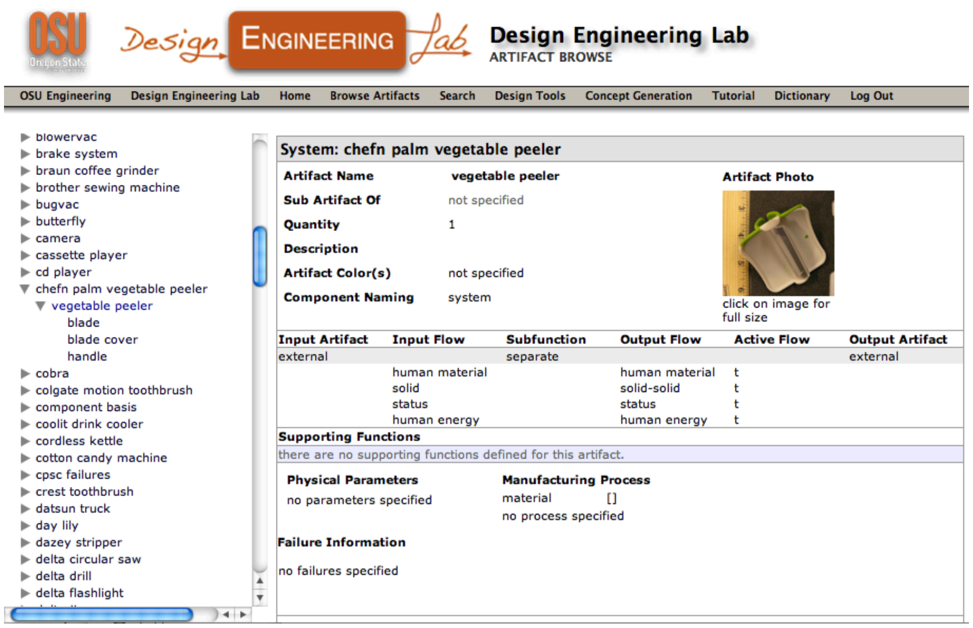


Figure 1.1 High Level Design Process

1.4.2 Design Repositories

Beginning in the late 1990's significant effort has been expended to develop design repositories that facilitate knowledge capture and reuse [10]. A design repository is a heterogeneous collection of product information that includes designer intent, solution principals, physical parameters, and models of products and their sub-artifacts. A design repository is distinct from more traditional design databases in that it records not only what has been designed, but also enough information to reason why and how the how artifact was created [11]. A repository also differs from an ontology which is a formal and explicit record of concepts and relationships [12]. Where the ontology records fundamental properties of artifacts and domains, the repository records artifact parameters necessary to reason about the underlying formal concepts. Consequently conclusions drawn from the repository evolve as the database grows. With an active and diverse group of contributors, a repository should tend to avoid systematic bias and correct errors. The use of a design repository as a data source distinguishes the proposed work from related efforts.

This work utilizes the prototype design repository maintained by the Design Engineering Lab at Oregon State University (http://repository.designengineeringlab.org). Figure 1.2 Design Repository Web Interface, depicts the repository's web interface. This repository was the result of a multi-university collaborative effort and is based on a NIST prototype [10, 13]. It is an artifact centric relational database populated with information from the disassembly and reverse engineering of existing products. The artifacts stored in the repository represent a diverse range of products from consumer goods to sub-systems of NASA spacecraft and even some biological systems. Data is input using a stand-alone entry application available from the repository webpage and is retrievable via either an online interface or via a direct query of the database [11, 14, 15].



- blowervac
- brake system
- braun coffee grinder
- brother sewing machine
- bugvac
- butterfly
- camera
- cassette player
- cd player
- chefn palm vegetable peeler
  - vegetable peeler
    - blade
    - blade cover
    - handle
- cobra
- colgate motion toothbrush
- component basis
- coolit drink cooler
- cordless kettle
- cotton candy machine
- cpsc failures
- crest toothbrush
- datsun truck
- day lily
- dazey stripper
- delta circular saw
- delta drill
- delta flashlight

System: chefn palm vegetable peeler

Artifact Name	vegetable peeler	Artifact Photo
Sub Artifact Of	not specified	
Quantity	1	
Description		
Artifact Color(s)	not specified	
Component Naming	system	click on image for full size

Input Artifact	Input Flow	Subfunction	Output Flow	Active Flow	Output Artifact
external		separate			external
	human material		human material	t	
	solid		solid-solid	t	
	status		status	t	
	human energy		human energy	t	

Supporting Functions

there are no supporting functions defined for this artifact.

Physical Parameters	Manufacturing Process	
no parameters specified	material	[]
	no process specified	

Failure Information

no failures specified

Figure 1.2 Design Repository Web Interface

1.4.3 Functional Basis and Component Taxonomies

The importance of functional decomposition of the design problem is a common theme across design methodologies and texts. The concept generation methods considered and the proposed new work both rely on the Functional Basis of Design [16, 17]. The Functional Basis defines a hierarchical taxonomy of function and flow terms. These terms are used to construct a black box model that represents the overall functionality of a product and its inputs and outputs of matter, energy and information. This black box model is subsequently decomposed into a series of sub-functions represented by a verb-object pair of function and flow from the Functional Basis taxonomy. The Functional Basis provides a systematic and repeatable framework for constructing functional representations of engineered products. Figure

1.3depicts a sample black box and functional model developed with the functional basis for a hand held electrically powered sander.

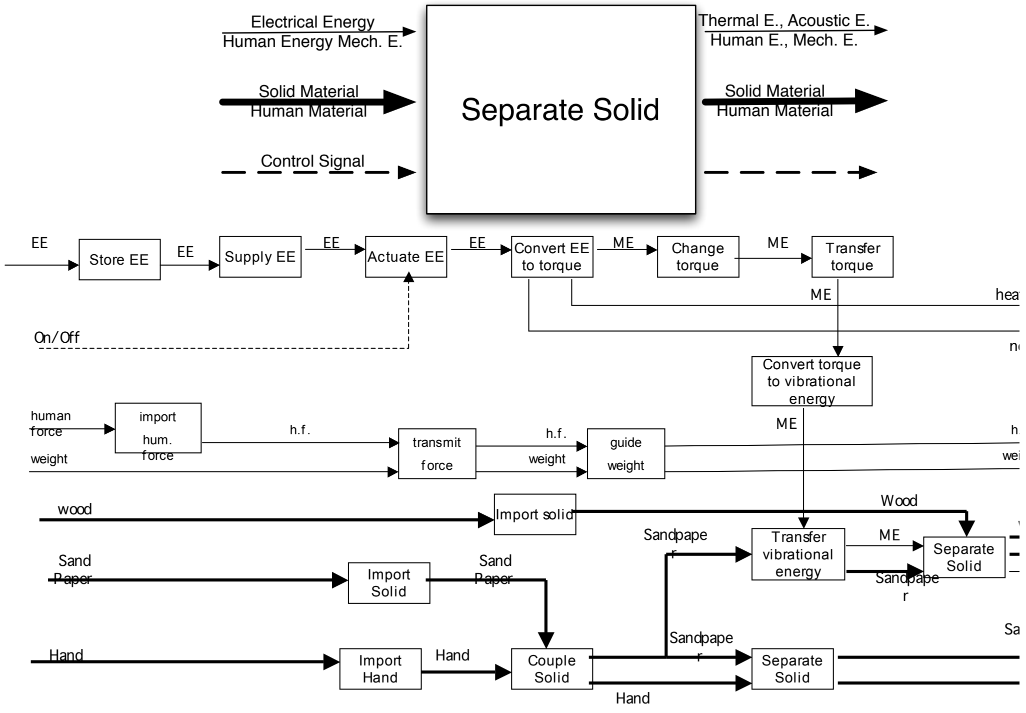


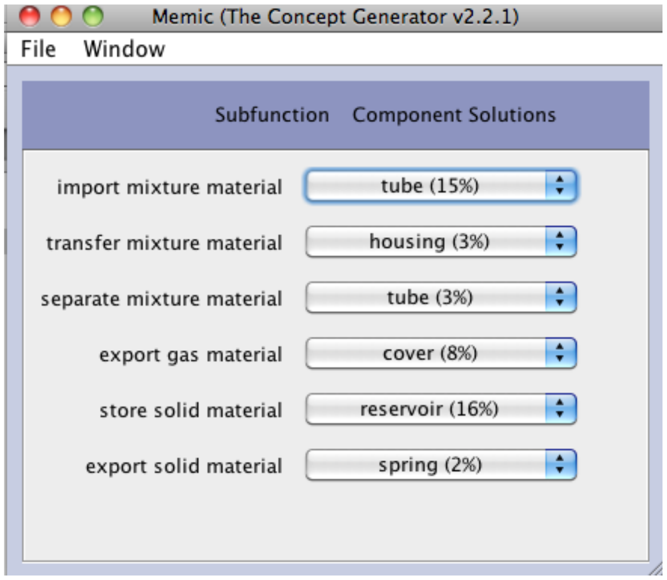
Figure 1.3 Functional and Black Box Models of Sander

In addition to the fixed vocabulary of function and flow terms, this work also relies on a fixed taxonomy of components. The need for such a taxonomy is highlighted by efforts to construct ontologies of design knowledge [18], but is also useful for computational design tasks. The taxonomy employed in this dissertation uses the reconciled set of component terms was put forward by Kurtoglu et al [19], and refined by Bryant [20]. The taxonomy organizes a set of archetypes of common electromechanical components into a hierarchy based on their typical functionality as described by the functional basis. Within this taxonomy 179 possible component types are recognized. Throughout this dissertation, the term component refers to one of these 179 recognized component archetypes.

1.4.4 Computational Design Synthesis

The proposed work falls into the broad area of computational design synthesis. This is an extensive and growing area of research, so a full review of cannot be accomplished in a work of this length. Instead this section will briefly outline the breadth of work in the field, establish a framework for understanding computational synthesis efforts, and survey literature related to the proposed work. Defined as "the algorithmic creation of designs [21]", this area of research emerged in the 1950's with computational tools for the design of electrical components including motors, generators, and transformers [22]. The goal of computational synthesis tools is to take advantage of the memory and computational capabilities of the computer to assist designers. Schon identifies four possible intents for computational design tools – functional equivalence to the human designer, phenomenological equivalence to the designer, aids to the designer, or as tools for design research [23]. It can be shown that a tool that is functionally equivalent to the human designer is a true artificial intelligence capable of passing the Turing test. Likewise, a phenomenologically equivalent tool would have to mimic the processes of the human designer; our understanding of these is so nascent that this too is unrealistic. Thus we are confined to trying to aid designers and support research efforts. Applications have been found in all stages of the design process.

Much work in the area of computational design synthesis has been devoted to the automated synthesis of product form by applying shape grammars. In particular efforts have used shape grammar techniques to build systems that automate architectural design work [24-26], and even mimicking the style of popular architects [27] or styles [28]. A slightly older, but still very relevant survey of shape grammars was conducted by Cagan [29]. Other work has applied the same kind of reasoning to automotive [30-32] and consumer products [33]. In a similar vein, Stahovich explored interpreting and modifying designer sketches to create new concepts [34-37]. Techniques utilizing a catalogue design approach represent some of the earliest application of computational synthesis techniques to conceptual design [38-40]. Other researchers have applied agent-based approaches to the problem of design synthesis seeking to computationally mimic the activities of a human designer. Notable among these is Campbell's A-Design which does an admirable job of applying itself to conceptual design problems, but seems at present restricted to problems defined by flows and transformations of energy [41, 42]. Others have developed approaches that rely on case-based reasoning [36, 43-45]. Welch developed a two step methodology which translates functional requirements to behavior and then to sets of components [46]. Recent work has sought to apply a grammar based approach to synthesize conceptual design solutions from functional problem descriptions [47, 48]. This work has been expanded to guide grammar rule selection based on designer preferences [49]. A potential shortcoming of this approach to using designer preference to guide generation is that it could lead to the very fixation problems that concept generation tools should ideally help alleviate. A limitation of all grammar-based approaches is the need to create grammar rules. Absent a method for discovering these natural through analysis of existing products, these rules will always be a snapshot of the experiences and biases of their authors. Finally, Bryant has developed MEMIC, a tool for automated concept generation utilizing the Functional Basis to create computable input functional models, a design repository as a data source and a derived component taxonomy [19, 50-52]. MEMIC was selected as the initial concept generator tool for the proposed research because evidence suggests that it enhances exploration of alternatives by novice designers, it is compatible with design repository data available to the researcher, and its use of models constructed using the Functional Basis. Figure 1.4 depicts the MEMIC tool.



Subfunction	Component Solutions
import mixture material	tube (15%)
transfer mixture material	housing (3%)
separate mixture material	tube (3%)
export gas material	cover (8%)
store solid material	reservoir (16%)
export solid material	spring (2%)

Figure 1.4 MEMIC Software UI

To frame an analysis of the MEMIC algorithm, the framework proposed by Cagan et al [21] and depicted in Figure 1.5 is adopted. This framework posits that there are four steps to the computational design synthesis task – representation, generation, evaluation and guidance. Representation is the process of transforming the problem statement into a format meaningful to the computer. Generation involves the computational tool creating a potential solution or solutions. Evaluation is the mechanism by which the computer determines if the generated solution meets the requirements of the original statement, and finally guidance is the process of directing the algorithm toward new solutions. These could either better meet the requirements of the designer or further explore the solution space. This trade-off between breadth and depth searching, between exploring all solutions and exploiting known solutions, is a key issue in all synthesis efforts.

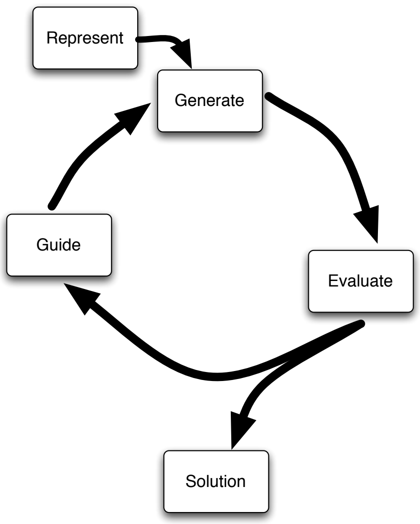


Figure 1.5 CDS Cycle adapted from [21]

If we apply the represent, generate, evaluate and guide framework to the MEMIC concept generator we find that it only partially completes the cycle. Representation is accomplished by functionally modeling the system to be designed and transforming this functional model into an adjacency matrix that the computer can interpret. Solutions are generated using an approach adapted from the morphological approach of Zwicky [53, 54] and the method of partial solutions advocated by Pahl and Beitz [5]. Evaluation is performed when the algorithm assesses component connection feasibility by consulting data in the Design Repository. The final step, guidance, does not exist in the current iteration of the tool. This requires the designer to handle the difficulties inherent in the combinatorial explosion of solutions produced. What is needed is an efficient way to assess concept similarity and guide the algorithm toward producing only the few novel variants in the set. The proposed research seeks to rectify these difficulties.

1.5 Conclusion

This dissertation explores a series of issues related to automated concept generation from functional descriptions of a design problem. Several approach are demonstrated which can enhance the utility of these tools by reducing the burden of evaluation and exploration placed on the user. These approaches rely on vector space models of design artifacts. Techniques from automated information retrieval are used to solve problems of similarity measurement among these vector space models. An existing function based design by analogy technique is used for benchmarking and validation. Guided by the results of these previous steps, the proposed techniques are extended to enable more intelligent automated concept generation that evaluates generated concepts and guides the generator to produce novel solutions.

2 Sorting Automated Concept Generator Output

2.1 Introduction

The following chapter attempts to improve the utility of automated concept generators without altering the underlying concept generation algorithm. This chapter begins by using the design repository data source, which is already an input to the concept generation process, to predict performance parameters of each concept. Designer workload while using the concept generator is reduced by extracting and presenting only those concepts with predicted performance above a given threshold. Noting that performance parameter predictions are correlated with component selection, a more general sorting approach is put forward based solely on the components of each generated concept. The size of the data set necessitates the application of tools for variable reduction to reduce the size of concept representations and exploratory data analysis. The results of applying these techniques suggest the direction taken in following chapters.

2.1.1 Roadmap

This chapter begins by introducing an approach to concept sorting based on estimating concept parameters. To achieve this a method for estimating the features of a concept from design repository data is put forward. The estimation method is demonstrated through two example implementations. Conclusions are drawn about this approach to concept sorting, and the implications lead to a new but related approach. Based on identified deficiencies of parameter based sorting, a new method which sorts based on component selection is put forward. This approach requires the introduction of variable reduction techniques and a method for automatically clustering data. Component based clustering is examined through three example products. Finally conclusions are drawn about this approach concept sorting.

2.1.2 Contributions of This Chapter

1. A method for estimating the parameters of conceptual products based on design repository data.
2. An approach to concept sorting based on parameter estimates is investigated.
3. A new method for concept sorting based on component choice is developed and applied.
4. Preliminary evidence that concept generator output is a set of permutations on a few solution types is found.
5. A first reduced vector space representation of automated concept generator output is introduced.

2.2 Concept Parameter Based Sorting

Chapter 1 introduced the notion of an automated concept generator and gave detailed background on a specific algorithm based on the morphological matrix. An observed drawback of this tool, and others, is that while a concept generator can produces hundreds or thousands of unique concepts, ultimately the user must evaluate and select concepts. Asking the designer to manually scan and parse thousands of concepts is an unreasonable expectation. If we imagine the role of the computer becoming more of a team member in the design process, this is not a good framework for fruitful collaboration. Currently, the problem is solved by asking the designer to interactively select components for each concept, which removes the need to explore thousands of concepts, but does not promote a thorough exploration of the design process. What is needed is a balanced approach which lets the concept generator do what it does best, generate many concepts, while freeing the designer to focus their energy on developing additional concepts and synthesizing the results into an outstanding product. The proposed method leverages the speed of the computer to eliminate a large number of undesirable concepts. A much smaller group of concepts can then be passed to the human designer whose reasoning and intuition can be employed to select from the reduced set.

2.2.1 Outline of Proposed Method

The output of the concept generator is a list of connected components, so a sensible approach to sorting concepts must be based at the component level. Given this constraint, there are then two general approaches; group concepts either by components used or by projected performance. It seems preferable to return concepts that will meet a designer's performance expectations, so an initial approach should be to select from a sample of concept generator output those solutions that are likely to meet some predefined targets. The variety in the set of solutions retained for presentation to the designer will depend on the number of different concepts with satisfactory components. In terms of the spectrum between exploration and exploitation discussed in Chapter 1, this method strives to exploit obviously workable solutions rather than explore the full breadth of the solutions space.

The general approach to sorting concepts will begin with selecting certain desirable or undesirable characteristics, determining the propensity for a particular component to have that characteristic, and then sorting concepts into groups based on the properties of their constituent components. The following diagram, Figure 2.1, summarizes the proposed method.

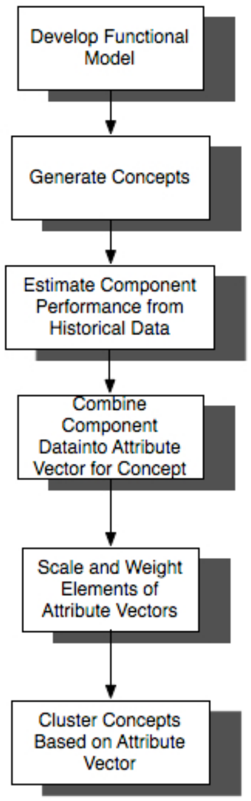

Develop Functional Model
Generate Concepts
Estimate Component Performance from Historical Data
Combine Component Datainto Attribute Vector for Concept
Scale and Weight Elements of Attribute Vectors
Cluster Concepts Based on Attribute Vector

Figure 2.1 Outline of Proposed Method

Functional Modeling and Automated Concept Generation were surveyed in the first chapter, so new work needed to implement this method begins with the third step, estimating component performance from historical data. In the next section, an approach for constructing these estimates and aggregating them to form a reasonable representation of each concept is discussed.

2.2.2 Estimating Concept Parameters from Repository Data

Each generated concept is represented as an adjacency matrix, essentially a graph of components and their connections that achieve the required input functionality. The basic assumption underpinning the proposed method is that a meaningful estimate of the performance of each concept, if it were built, is a function of the expected parameters of its components. A second basic assumption is that the parameters of the components that would make up the concept can be estimated by surveying components of the same type stored in a design repository. A necessary precondition is a fixed component vocabulary like the one discussed in Chapter 1.

Various earlier research efforts have used design repository data to estimate the behavior of products during conceptual design. A significant body of work is centered on estimating the likely failure modes of a product and it's reliability. Some of this work has used data stored in a repository to discover links between product function and product failure that can be used to predict failure modes in future products [55, 56]. This work has been refined to focus on high-risk product domains like aircraft and spacecraft [57], and extended to include even software development [58]. Much of this work has been refined into a framework for understanding and predicting failure propagation in complex systems in early design [59]. A parallel effort has investigated predicting not just failure modes, but their corresponding likelihood and severity [60]. This failure estimation work inspired attempts to estimate other product parameters from repository data. Parashar et al used this data to estimate the part count of a product in early design based on existing products [61], while Poppa attempted to estimate product manufacture and assembly costs [62]. A generalized version of that approach forms the basis of the following method.

Estimating concept parameters begins with selecting a set of parameters to estimate. While no formal rule can be devised for this, experience dictates some best practices. First, parameters should be directly related to important product requirements, and should be directly related to the evaluation scheme that will be used for concept selections. Thus parameters should be relatable to function or component type because repository data is keyed those values. For example, estimating color would be fruitless because we have no reason to suspect that knowing a concept's functionality or the parts that make it up will tell us what color it will be. Finally, these parameters should be generalizable to all components, though the estimation scheme could be component or function specific. For example, number of gear teeth, is data that can be extracted for particular instances of gears in the repository, but it's not suitable for describing a variety of components. A better choice might be the input/output ratio of components that transfer energy.

To predict the parameters of an individual component in a concept, the design repository is queried for other representative components of the same type. There two cases to consider. In the first, the parameter of interest is part of the repository data schema and is potentially recorded for each component. In the second, the parameter of interest is not directly available. If it is a function of parameters that are captured, then estimates can proceed as in the first case. If it is not a suitable proxy must be found, more data must be collected, or a different parameter selected.

Representative components of the same type are a somewhat nebulous concept, and as with selecting parameters, no single rule will suffice for all cases. Some important concerns are scale, product domain, operating environment, and function-component interactions. Scale is an obvious issue. Most repository artifacts are small-scale consumer electromechanical products. A few are not, and those that are wildly different from the scale of the product being designed, should be excluded from estimates. The same is true for product domain and operating environment. Predictions about a prospective kitchen appliance should not utilize data from spacecraft subsystems as its requirements are so far afield that choices about materials, process, size, and other attributes are unlikely to be representative. The reverse is probably also true. Finally, there is the issue of whether components of the same type, but which were observed performing a different function should be included. Requiring both function and component to match will necessarily lead to smaller samples. The specific application will dictate the appropriate course of action, but for some product parameters estimates based on both function and component may improve results. Failure modes for example are related to both what a component is, and what it does. On the other hand, for a particular class of components, we might conclude that material selection is primarily a function of component type irrespective of function.

Once a sample of components has been found the properties of the components used in the generated concepts must be estimated. For various problems these can be estimated as the mean of the samples, an extreme value, or a range of values. Once this estimate is found at the component level, these values can be combined to estimate the parameters of the assembled concept. The parameter being estimated, the particulars of the design problem, and sound engineering judgment will determine the appropriate way to do this. Summing the components, taking an average or weighted average, and using the most extreme value have all been successful for various problems.

Combining estimates of several parameters introduces a vector space representation of a concept's expected performance along dimensions the designer has declared important.

Once the data and estimation procedure are in place, asking a computer to perform the necessary calculations for an available sample of concepts is straightforward. Care should be taken in interpreting the results; there will necessarily be significant uncertainty in the results. A direct rank ordering is both unreliable and misleading. Instead, concepts with similar performance estimates can be grouped together and groups with poor performance can be rejected while those more likely to meet designer expectations can be passed to the designer for further evaluation.

The following sections take this general template for concept performance parameter estimation from a repository of existing products and apply it two specific cases. In the first, estimates are made to assess the manufacture and assembly cost of automatically generated concepts. In the second an attempt is made to extract information necessary for product environmental impact estimates.

2.2.3 Sample Implementation 1: Design for Manufacture and Assembly

To better illustrate the proposed method, the following example is presented based on sorting for Design for Manufacture and Assembly (DFMA). A set of sorting parameters could also be specified based on another set of design for "X" rules, from a set of customer needs, or from any combination of those. The intent is simply to show what an actual implementation of the method might look like. Four characteristics are identified based on DFMA concerns in consultation with the standard DFMA literature [63]. It is assumed that the product being designed was a high volume small-scale consumer product. Use of standard or OEM parts, use of low cost thin walled stamped and injection molded parts, and avoidance of machined components were selected as useful parameters for estimation.

It is assumed that concepts should be built from standard parts to the greatest extent possible. Standard parts are common, standardized components that can be sourced from a variety of suppliers. They might alternatively be referred to as off the shelf, or original equipment manufacturer (OEM) parts. The basic assumption is that market forces will drive the price of these standard parts below the cost of the manufacturer to make a custom part [63].

To estimate the degree to which a concept can be built using standard parts, it is first necessary to associate some measure of standardness with each term in the component taxonomy. This is an example of a case where the necessary data is not directly stored as part of the repository schema. However, it was concluded that standardness was a property that could be assessed visually, and artifacts do have images associated with them. All artifacts tagged with a component taxonomy name were extracted from the repository. Using artifact photos and other data the artifact was tagged as either standard or not standard. The binary nature of the data implied a Bernoulli distribution for each artifact, from which could be estimated a mean, variance, and confidence intervals for the mean. The findings matched expectations. Electric motors, which are almost universally standard parts, have a high average, while external housings, which tend to be custom parts to accommodate product architecture as well as branding and aesthetic concerns, have a very low average value.

Once the expected standardness for each component is known, a measure of standardness for the concept as a whole could be generated. There are two approaches to estimate component standardness. The first would be to multiply together the standardness of each component in the concept to produce an estimate of the likelihood that the entire concept could be made from standard parts. The likelihood that a concept made of components A and B is standard is the likelihood that A and B are standard parts. This would be preferred because in reality all components should have some non-zero standardness, and the aggregate of these would be a reasonable representation of our ability to build the concept from standard parts. In this case, due to small sample sizes of some components in the repository many of the current estimates of mean standardness are zero. This would strongly penalize an otherwise desirable concept due to one non-standard component. To mitigate this effect standardness of the concept was estimated by simply averaging the standardness of the components.

Second, it is deemed desirable to use thin walled parts produced using stamping or injection molding. These processes are common and relatively low cost for the high volume parts produced for consumer products. These processes are likely to be selected, so concepts that include components which can be produced at a low cost with these methods are preferred [64].

It is also assumed that it is desirable to use thin walled parts produced using stamping or injection molding. These processes are common and relatively low cost for high production volumes. To estimate the relative cost of these processes for each component taxonomy term, the database was queried to find all artifacts produced using one of these processes. Unfortunately, sample sizes were prohibitively small for stamped parts. This highlights a disadvantage of the proposed method. Sometimes necessary data will not be available for key parameters.

For each artifact a cost estimate is then calculated. A method for cost estimates that relate the cost of the part to the cost of a standard unit washer is employed [63]. The cost becomes a product of several factors based on the parts features, parts, and materials. Excluding factors related primarily to material or fit and finish, left basic and subsidiary part complexity to base a relative cost upon. Using DFMA heuristics these factors were calculated for each artifact, and multiplied together for a total relative cost. These costs were then averaged for each component within the taxonomy. A thin walled part cost factor for each component was calculated by averaging the relative costs of its thin walled parts.

Third, machining is a wasteful and costly process that should be avoided if possible. Concepts built from components that are unlikely to require machining were preferred. Machining is a costly and undesirable process for components within the specified product domain [63, 64]. The likelihood that a particular component taxonomy term will have to be machined can be estimated by querying the database for all artifacts of a particular component basis type that are machined and dividing by the total instances of that component basis type that have any manufacturing process associated with them. Dividing only by instances that have a specified manufacturing process prevents incompletely recorded artifacts from heavily swaying the results. The results conform to basic expectations about this domain; the likelihood that a part will be machined is low for all component taxonomy terms.

Finally, a proxy for assembly cost is needed. The form of the individual solutions and the way they are joined together embodies much of the assembly cost, but this is a relatively complex relationship. As advocated in the previous section, a suitable proxy had to be found. In this case, total part count was selected because assembly cost is, all other things being equal related to the number of pieces that have to be joined together.   Parashar [61]has developed a part counting tool that interfaces with the design repository to produce an estimate of the number of parts necessary to complete the concept. The counter is based on the average number of instances of a component found when that component is used to solve a specified function [61]. The count produced by Parashar's tool became a fourth metric used to sort components.

Once the characteristics of each concept have been found, they will be combined, following it prescribed method, into an parameter vector that suggests the manufacturability of the concepts. Concepts can then be sorted into groups based on their similarity to one another.

The four calculated product parameters are assembled into a vector representing the relative manufacturability of the concept. The designer wishes to minimize three parameters: thin walled part cost, likelihood of machining, and part count. The number of standard parts, on the other hand, should be maximized. For convenience the standard part likelihood is transformed into a not standard part likelihood by subtracting it from one. Now all dimensions should be minimized. A sample calculation of an parameter vector for a concept including a electric wire, an electric switch, and a battery is shown in Table 2.1. To avoid undue weighting of a particular parameter all are normalized on a scale from 0 to 1. The normalized product vectors can then be clustered using manual or automated clustering techniques. A full discussion of automated clustering algorithms is reserved for latter in the chapter, but the approaches discussed there are also suitable for the product parameter vectors shown in this example.

Table 2.1 Parameter Vector for Sample Three Component Concept

Sample Concept				Parameter Vector
Components	Battery	Electric switch	Electric wire	
Standardness	0.6750	0.3874	0.8839	0.6488
Machining Likelihood	0.0000	0.0078	0.0051	0.0043
Thin Wall Relative Cost	na	2.0772	na	2.0772
Part Count				7.3559

To demonstrate concept variant sorting method, concepts for a children's toy will be developed. It is desired that the toy translate across a surface using stored electrical energy, and that it be very low cost to produce. A functional model for the toy generated is shown below in Figure 2.2.

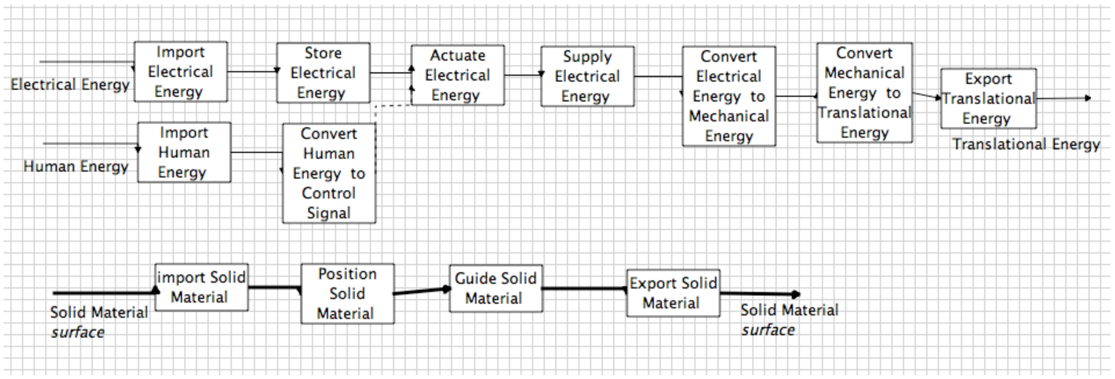
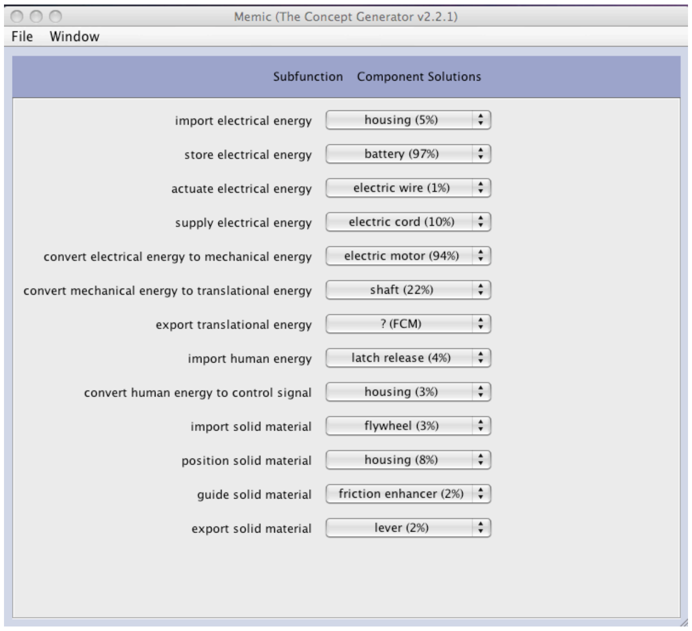


Figure 2.2 Functional Model for Case Study of Toy

The functional model was input into the MEMIC concept generator along with a FCM and DSM Matrix from the design repository. A sample of the results is shown below in Figure 2.3. Many concepts are produced, but in the interest of producing an understandable set, twenty-five are selected at random for further review.



Subfunction	Component Solutions
import electrical energy	housing (5%)
store electrical energy	battery (97%)
actuate electrical energy	electric wire (1%)
supply electrical energy	electric cord (10%)
convert electrical energy to mechanical energy	electric motor (94%)
convert mechanical energy to translational energy	shaft (22%)
export translational energy	? (FCM)
import human energy	latch release (4%)
convert human energy to control signal	housing (3%)
import solid material	flywheel (3%)
position solid material	housing (8%)
guide solid material	friction enhancer (2%)
export solid material	lever (2%)

Figure 2.3 Sample Concept Generator Output

Parameter vectors for each component can be calculated by following the algorithm discussed at the beginning of the section. With only twenty-five concepts, it is possible with effort to identify preferred concepts. If the set were larger the computer's assistance would be needed. The vectors are supplied to automated clustering algorithm that suggests dividing the results amongst four clusters. Table 2.2, shows the parameter vector and cluster membership of each concept.

An examination shows that cluster three's center is closest to the origin. Its members have relatively low combinations of costly parameters. Concepts 2, 5, 7, 19, 20, 21, and 24 likely merit further study based on DFMA concerns.

A further examination of these concepts suggests that they fall into two general categories. Given the very general approach used to model the problem some interpretation is required. Concepts 2, 5, 7, and 21 imply a walking toy, like a toy robot. Concepts 19, 20, and 24 suggest something more like a toy car. Based on these results the designer would have a few concepts suggesting two different solution types to work with.

Table 2.2 Results for Toy Case Study

Concept	Std Pts	Machining	Thin Wall	Pt. Count	Cluster
Concept01	0.8240	0.9768	0.8089	0.6413	4
Concept02	0.8012	0.9597	0.5250	0.5725	3
Concept03	0.6587	0.9910	0.8621	0.7405	4
Concept04	0.9337	0.9935	0.6385	0.7743	1
Concept05	0.8253	1.0000	0.6037	0.5583	3
Concept06	0.7309	0.9762	0.6230	1.0000	2
Concept07	0.7430	0.9452	0.6400	0.6996	3
Concept08	0.8912	0.9504	0.6083	0.7537	1
Concept09	0.6449	0.9794	0.8063	0.7614	4
Concept10	0.8086	0.9921	0.8357	0.8178	4
Concept11	0.9691	0.9862	0.6258	0.6167	1
Concept12	0.7795	0.9760	0.8521	0.7361	4
Concept13	0.5976	0.9492	1.0000	0.8117	4
Concept14	0.9123	0.9690	0.6292	0.7097	1
Concept15	0.6792	0.9462	0.7389	0.7278	4
Concept16	0.9739	0.9858	0.8253	0.8780	1
Concept17	1.0000	0.9829	0.7858	0.7564	1
Concept18	0.7210	0.9972	0.8164	0.5806	4

Concept19	0.8443	0.9953	0.7050	0.6192	3
Concept20	0.7123	0.9512	0.6793	0.3992	3
Concept21	0.7602	0.9637	0.6818	0.5926	3
Concept22	0.6651	0.9955	0.8994	0.6376	4
Concept23	0.5653	0.9668	0.8615	0.7006	4
Concept24	0.7856	0.9645	0.6030	0.5794	3
Concept25	0.6502	0.9835	0.8491	0.8394	4

The results of the case study provide preliminary evidence supporting the hypothesis that product parameters can be estimated basted on data in a design repository and successfully used to pare down the set of results returned by an automate concept generator.

2.2.4 Sample Implementation 2: Estimating Data for Environmental Impact Assessment

Another potential application of the proposed sorting method is to select concepts that will have minimal environmental impact. This approach has contributed to recent literature in artificial intelligence in sustainable design [65, 66]. In the previous sample implementation, estimates were made to predict manufacture and assembly cost for concept screening. In this effort, an estimate of environmental impact is needed. This is not a value than can be directly predicted via past products; environmental impact estimates are not currently part of the repository data schema. However an examination of a common environmental impact estimation tool, shows the that dominant factors in impact are material selected, mass of material, and manufacturing process employed [67]. The proposed method estimates these values necessary for estimating the cradle to gate life cycle impact of the generated concepts based on data stored in the repository.

Each component of products archived in the repository is tagged with the material or materials from which it was constructed. The material of the component used to build the concept was predicted from this set. Mass and volume data are recorded for artifacts in the repository, so an estimate of both the type and amount of material used is possible. A designer could specify, based on their application, whether to estimate the material based on a best, worst, or average case material, but experience has suggested that creating an estimate based on a hypothetical composite material that is a weighted average of the most common materials found for a given component gives good results [65]. For example if we found that 80% of instances of a particular component were brass and 20% were nylon, the hypothetical component impact would be estimated based on a part that was 80% brass and 20% nylon by mass.

Estimation of manufacturing process proceeded in a similar fashion, but with a key added complications. The estimated process must be appropriate for the material estimated in the previous step. Thus, we estimate the likely manufacturing processes found for components of the specified type with the predicted material. This process can also work in reverse. The process can be predicted first, and then the material choice constrained accordingly. In this case, the former was selected because of greater confidence in the material data within the repository than in the manufacturing process data. This work contributes to environmental impact assessment in early design by providing a method to extract needed data from a design repository.

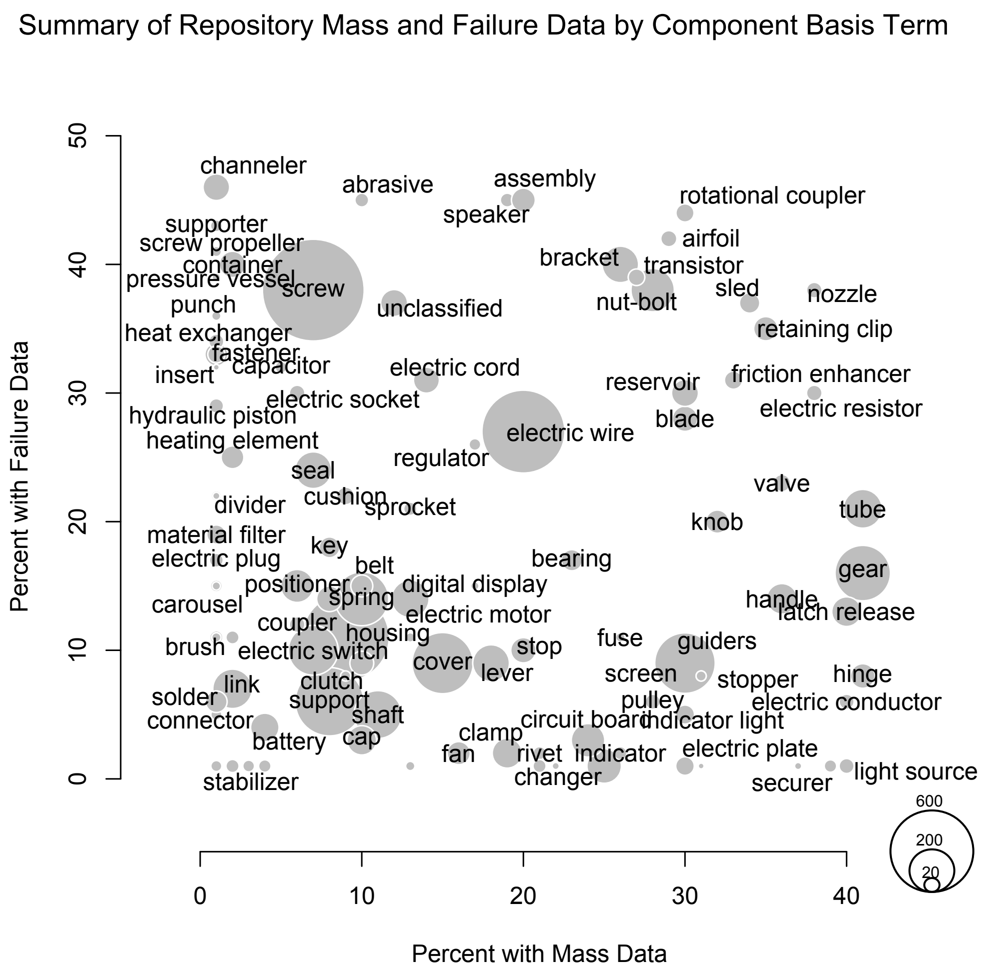


Figure 2.4 Summary of Available Mass and Failure Data

2.2.5 Conclusions about Parameter Estimation Based Sorting

Sorting concepts based on predicted concept parameters or performance clearly presents some challenges. While it was possible to employ this approach in the two samples discussed in the preceding sections, a great deal of effort had to be extended at the beginning of each problem to obtain and organize the data necessary to make meaningful estimates. The quality and the reliability of the results are heavily dependent on the quality and quantity of data in the repository and on the estimation model constructed by the user. To illustrate this point consider Figure 2.4 a plot that shows samples sizes for common components in the taxonomy for mass an failure data. These are commonly relied upon results and yet all have very small samples. This trend only gets worse for less common and less easily collected data.

It is also inefficient to calculate estimates for many concepts that will ultimately be rejected. While this is a partial solution to the difficulties of automated concept generators discussed in chapter one, an easier and less problem dependent approach is desirable. In applying this approach to a variety of problems, an important trend emerges that perhaps should have been obvious from the beginning given the way parameter estimates are constructed: Concepts with similar estimated parameters tended to be composed of the same components. Based on that observation, I hypothesized that those concepts could be meaningfully sorted into groups based solely on their components. Then, if necessary, estimates could be made based on samples from each group. The following section outlines this approach.

2.3 Sorting Based on Component Selection

The previous section notes that concept parameter estimates tended to lead to concepts constructed of similar components being grouped. The logical step is to omit the estimation of parameters and simply group concepts together based on the components from which they’re assembled. The ultimate goal is to construct an algorithm that can read a large number of automatically generated concepts and sort them into bins based on their similarity to one another. If we consider that the concept generator is to a large extent a more advanced version of the morphological matrix, a logical hypothesis might be that the concept generator output is really many permutations on a few solution types. The total number of solutions is constrained by the number of possible solutions to the products defining functions. This is a hypothesis that will be evaluated through case studies later in this section.

For an algorithm to sort concepts, they must be represented in a computable form. In the previous section concepts were eventually represented as a vector of parameter estimates. The component adjacency matrices, which are the output of the concept generator, can be thought of as comparable to these concept vectors. However, in this case entries in the vector represent component choice rather than concept parameters. Since we prefer to operate on concept vectors rather than concept matrices, each adjacency matrix can be converted to an adjacency list by simply appending successive columns below the first. The result is vector whose elements represent the presence or absence of a particular component-to-component connection in the concept. Though sparse, the length of these vectors is an issue. Recall from Chapter 1 that the number of component types in the component taxonomy is 179. So an adjacency matrix that includes all possible components would be 179x179, and a corresponding adjacency list could have up to 32,041 elements. By omitting components that are not present in any of the generated concepts, the size of this list can be reduced, but in general it will still be large. Techniques to reduce the size of this matrix will be discussed, but first a computational approach to grouping concepts is briefly reviewed.

2.3.1 Cluster Analysis

Given a set of concepts represented as adjacency lists of components, the task is to sort concepts into groups based on component similarity. Measuring the distance between concepts is simple arithmetic, but automatically sorting them into groups based on their distance to one another requires the use of a set of techniques called clustering or cluster analysis. Clustering is the general term for a set of exploratory data analysis techniques used to solve grouping or classification problems [68]. Clustering methods tend to be heuristic in nature, and are most applicable when there is little information about the underlying structure of the data [69]. The objective of clustering analysis is to sort a set of data into groups, or clusters, such that each member of a cluster has a high degree of similarity with any other member of the cluster and a low degree of similarity with any non-member of the cluster [68]. There are two general types of clustering algorithms. Hierarchical clustering looks for a series of nested partitions in data. Partitional clustering, as the name implies, calculates a single set of partitions. In either case the fundamental problem is to sort n observations in d-dimensional space into K groups based on a specified similarity criterion [68].

In this work we utilize a variation of the K-means algorithm. K-means is an iterative clustering algorithm that assigns observations to a specified number of clusters, K. First K cluster centers that evenly span the space of observations are used to create an initial partition. Each observation is then assigned to the closest cluster center and new cluster centers are calculated based on the centroid of each group. This process is repeated until the square error is minimized and the cluster membership stabilizes [68]. Unfortunately, clustering is computationally expensive; we can make it more tractable if the number of dimensions that represent each concept can be reduced. The following section discusses an approach to variable reduction that can achieve this result.

2.3.2 Variable Reduction Via Principal Component Analysis

Principal component analysis (PCA) is a common method for dimensional simplification in multivariate data. PCA forms a new set of variables, the principal components, which are linear combinations of the original variables. These new variables form an orthogonal basis for the original data. Orthogonality is an important property; it ensures that no redundant information is created in the variable transformation.

There are many ways to form an orthogonal basis of a data set. In PCA each component represents an axis in the data space, and projecting the data onto this axis forms a new variable. PCA repeatedly performs this transformation, at each turn selecting the axis that results in a new variable that explains as much of the variance in the original data as possible. A key set in this process is the singular value decomposition of the original data matrix. Singular value decomposition is an important feature of a methodology that will be introduced in chapter 4, so a full discussion of it can be found there. For now, we will assume that PCA represents the best affine transformation of our original data matrix.

Though the full set of principal components is as large as the set of variables in the original data, typically a few components account for the majority of variance observed in the original data. It has been demonstrated that by representing the data set with this reduced set of components, which captures the majority of observed variaince, we can significantly reduce the number of individual variables under consideration while still capturing sufficient design information [70].

2.3.3 Case Studies of Component Based Clustering

The following section explores a component-based approach for automatically sorting the output of a concept generator into meaningful groups. We investigate this technique through the use of three sample problems from prior work [71]: a product to automatically remove the shells from peanuts as an aid to farmers in the developing world; a device to move fluid from a reservoir at one elevation to another at a higher elevation; and a consumer product to grind and dispense whole spices. Figure 2.5 shows a graphical representation of the analysis procedure utilized.

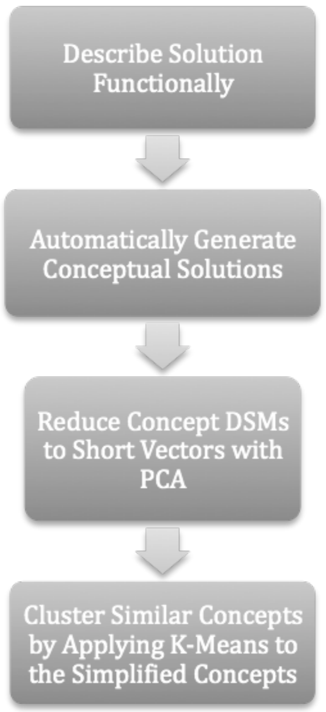

Describe Solution Functionally
Automatically Generate Conceptual Solutions
Reduce Concept DSMs to Short Vectors with PCA
Cluster Similar Concepts by Applying K-Means to the Simplified Concepts

Figure 2.5 Proposed Component Based Sorting Method

A functional model was constructed for each of the three sample problems based on known requirements and customer needs input. These functional models were then supplied to a concept generator. For each problem the concept generator was asked to supply one thousand concepts at random. Concepts were output in the form of 179x179 element component adjacency matrices representing all possible combinations of members of the component taxonomy.

Each component adjacency matrix was then reformulated into a vector by appending each column of the matrix to the preceding column to produce an 18,496-element vector where each element represents a possible connection between component types. These vectors are then aggregated into a matrix where each column represents a concept and each row a possible component interaction. For the sake of efficiency, rows of the matrix that contain only zeroes were deleted; if a particular component to component connection is never present in any concept there is no value in retaining that variable in the model.

PCA was applied by treating each concept as an observation and each component interaction as a variable. The number of principal components to retain was determined by producing a scree plot of the eigenvalues of the principal components and selecting components from the steepest portion of the curve. The more precise Kaiser Criterion [72] was also investigated, but found to select too few principal components to adequately reproduce the original data. Once the number of principal components is identified, the original data is transformed onto the principal component space, and a partitional clustering technique can be used to group concepts.

Once the concepts have been reduced by PCA into vectors of a few components, a standard K-means clustering algorithm was employed. An initial partition was created by selecting K cluster centers that evenly spanned the space of observations. Each observation was then assigned to the closest cluster center and new cluster centers were calculated based on the centroid of each group. This process was repeated until the square error is minimized and the cluster membership stabilizes. After a solution was found, the appropriateness of the number of clusters chosen was evaluated quantitatively by examining the mean square error between cluster centers and qualitatively by examining silhouette plots of the data. A silhouette plot shows the ratio average distance between a point and the other members of its cluster to the average distance from the same point to the members of the next closest cluster [73]. High silhouette values indicate strong affinity for the current cluster, low values indicate the point has been misclassified. The number of clusters K was iterated until silhouette plots show the clusters are well differentiated and the mean square error between cluster members and cluster centroids is acceptably low.

The method was first applied the concepts generated for the nut shelling problem. The 1000 concepts were supplied to the PCA algorithm. From the following scree plot, shown in Figure 2.6 Scree Plot of Peanut Sheller Results, it is apparent that the correct number of principal components to retain is approximately six. The scree plot is used to graphically assess how many principal components should be retained and is related to the eigenvalues of the principal components. The number of components to retain is the location on the horizontal axis that corresponds to a leveling of the curve's slope. Six principal components are selected, but it is worthwhile to investigate neighboring values. No significant difference in the results was found by increasing the number of principal components (PC) between six and ten. However decreasing the number to five or fewer led to cluster instability and suggested that there was too little differentiation between concepts.

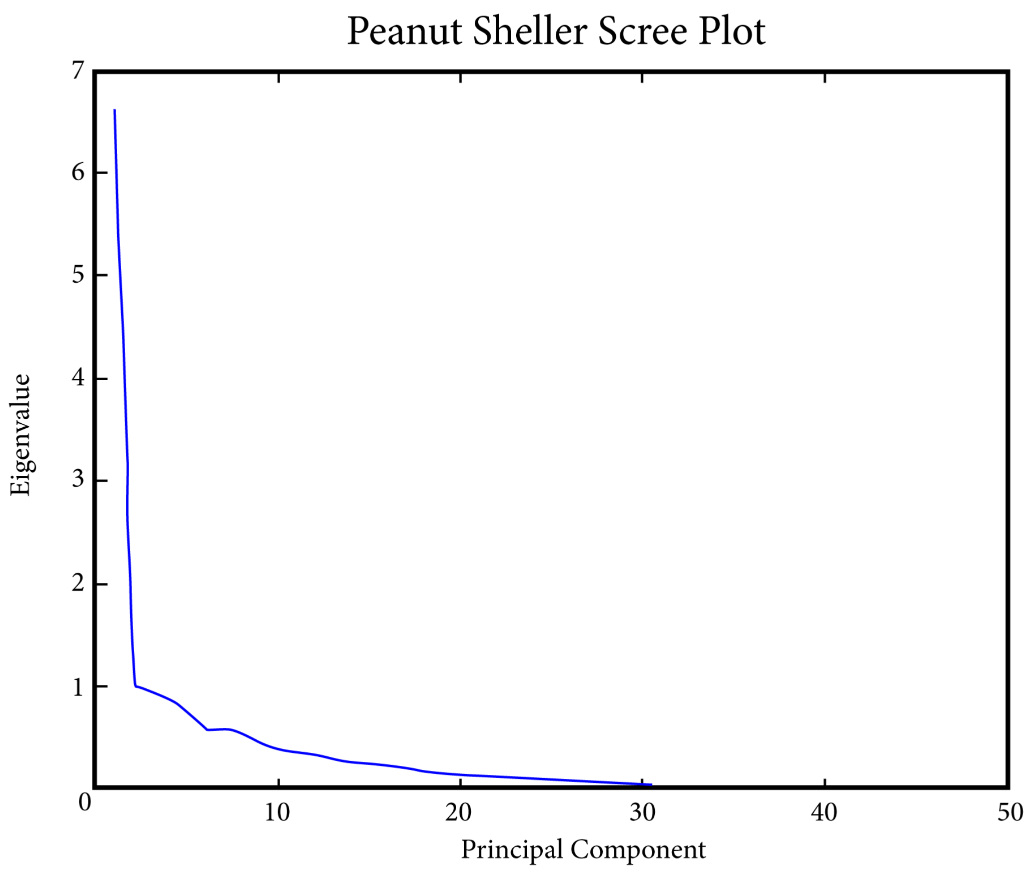


Figure 2.6 Scree Plot of Peanut Sheller Results

Next, the raw data is transformed onto the PC space and is clustered using the Euclidian distance between concepts as a similarity metric. Several possible numbers of clusters from two to fifteen were considered, but two clusters were found to minimize mean square error and give the most satisfactory silhouette plot as shown in Figure 2.7.

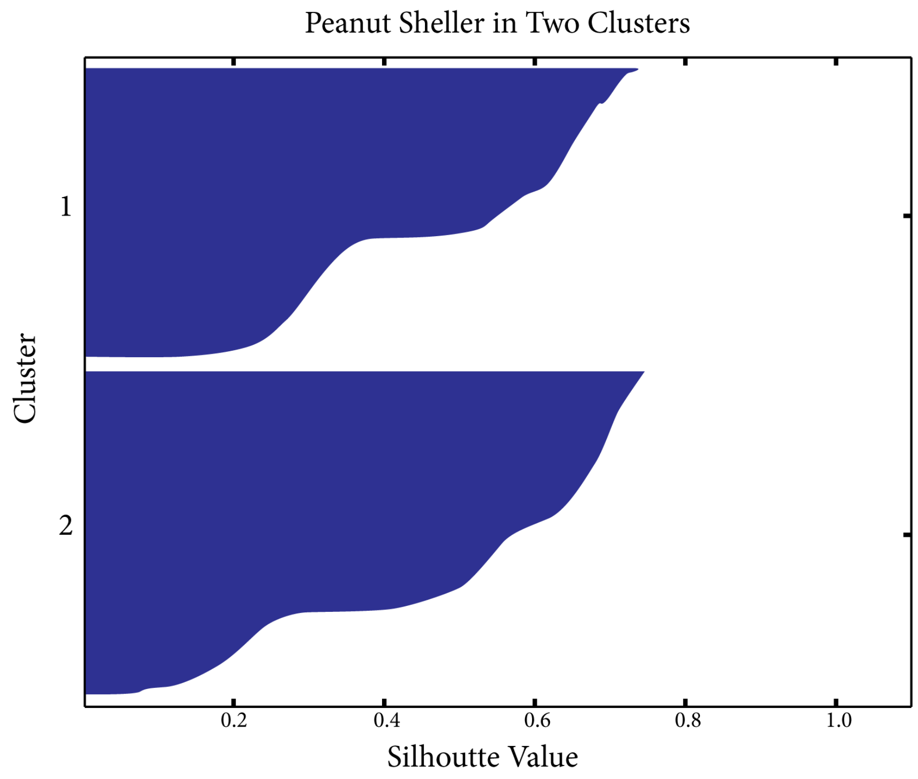


Figure 2.7 Silhouette Plot of Peanut Sheller Concept in Two Clusters

An examination of the plot shows that both clusters have some members with low silhouette values, but experimentation showed that this could not be improved through increasing the number of clusters. In general there appear to be two types of components that, by examination of individual concepts, could be broadly described as products that cut the shell and products that crush or grind the shell. The low silhouette values appear to indicate concepts that do both but are similar enough to one or the other that they do not form a cluster of their own.

Next the approach was applied to the water lifter functional model. As with the previous problem the each of the automatically generated concepts are supplied to the PCA algorithm. By examining the resultant scree plot, shown in Figure 2.8, the appropriate number of PCs to retain is found to be five.

However through investigating the results of selecting neighboring numbers of PCs it was found that results changed noticeably as the number of PC's increased to ten. In light of this ten components were retained. This discrepancy is due to the imprecise nature of using the scree plot to select the number of principal components, and highlights why it is important to investigate values in the neighborhood number suggested by the plot.

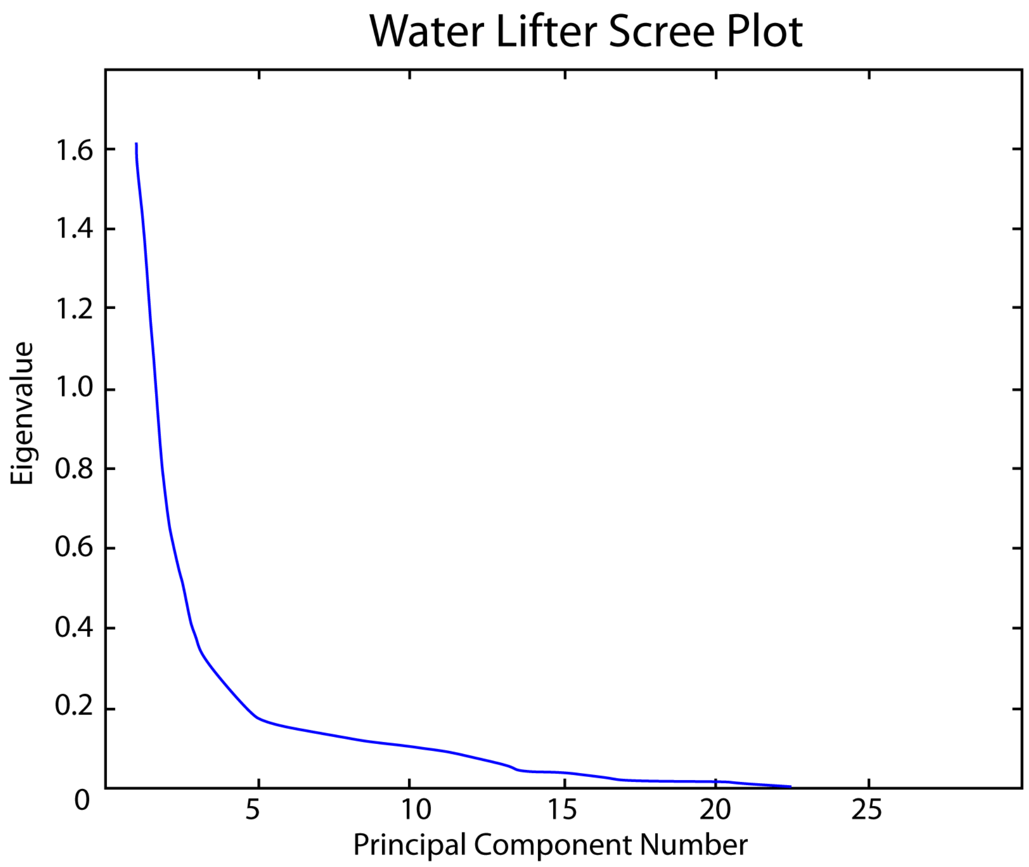


Figure 2.8 Scree Plot of Water Lifter Results

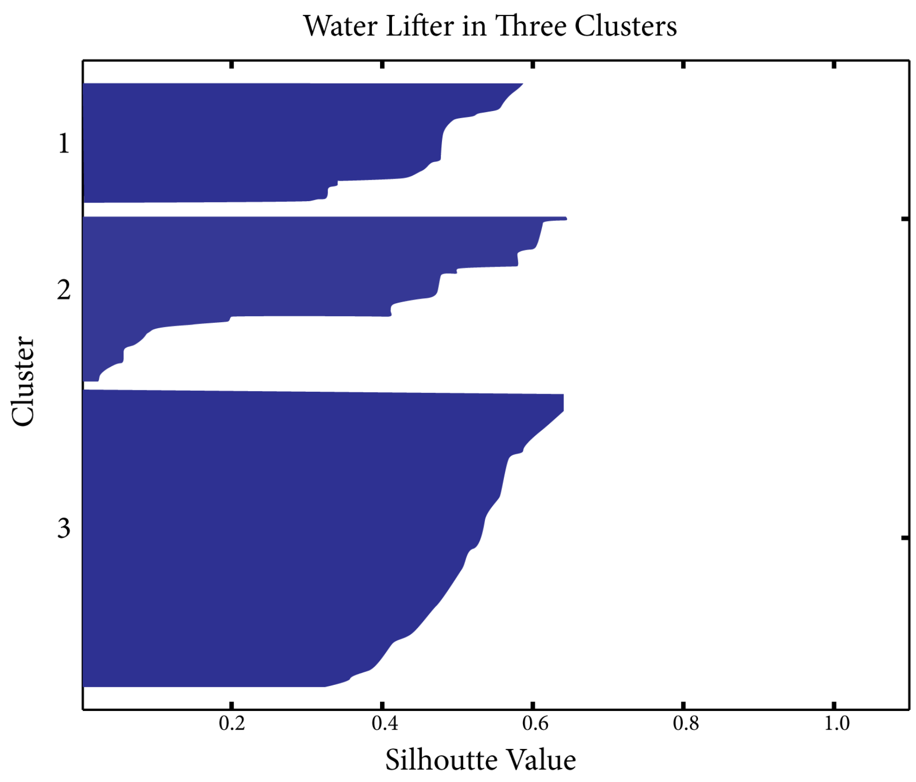


Figure 2.9 Silhouette Plot of Water Lifter in Three Clusters

Using the selected number of principal components, the data is transformed onto the PC space, and as in the previous example, is clustered using distance as the similarity metric. Again, a variety of possible divisions were investigated, but three was found to be the most appropriate number of partitions. The following figure shows the silhouette plot of the results for three clusters.

In this example, most concepts belonged to one large cluster with some concepts belonging to each of two smaller clusters. The clusters were better differentiated, as indicated by higher average silhouette values. However, when individual concepts were examined it was difficult to discern what the clusters represented. In general it appeared that the big cluster represented concepts that directly convert an energy input into the flow of water, while the other smaller clusters represented solutions with a number of intermediate energy conversion and transmission steps.

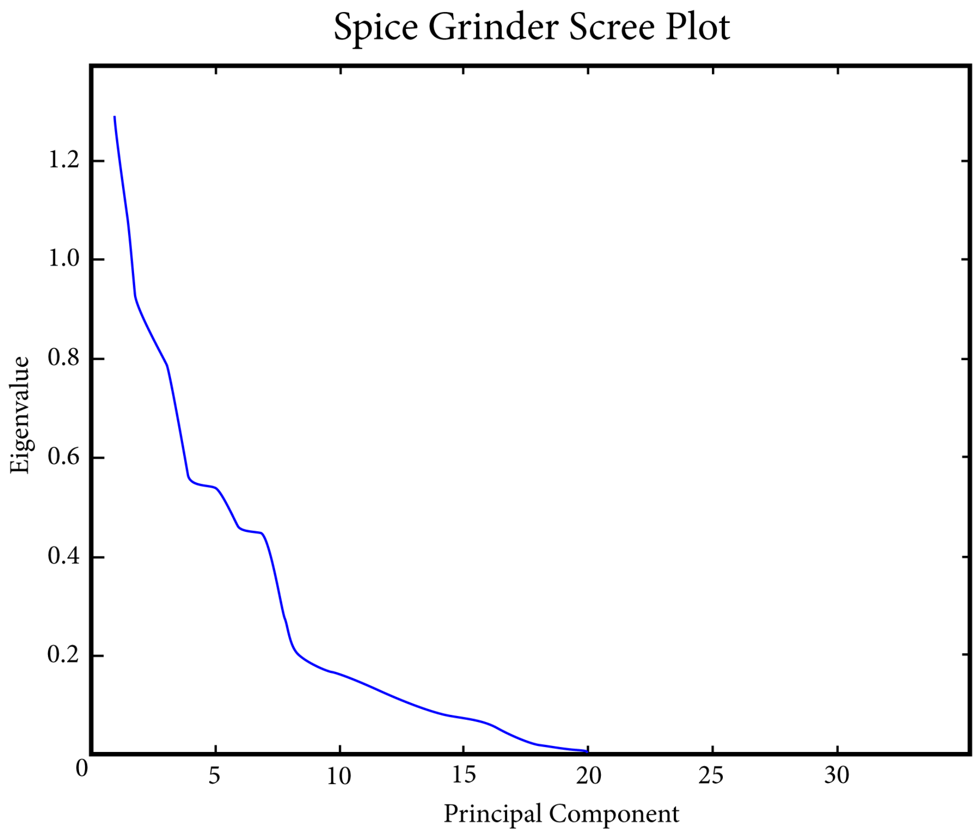


Figure 2.10 Scree Plot of Spice Grinder Results

Finally the suggested approach was applied to the spice grinder problem. Figure 2.10, was used to select the appropriate number of clusters to retain. An examination of the plot indicated that the appropriate number of components to retain was between six and eight. Investigating the choice in the neighborhood of each, eight was found to be the appropriate number of components to use.

After applying K-Means clustering iteratively to the variable reduced concepts six partitions were found. The following figure, Figure 2.11, is a silhouette plot of the results.

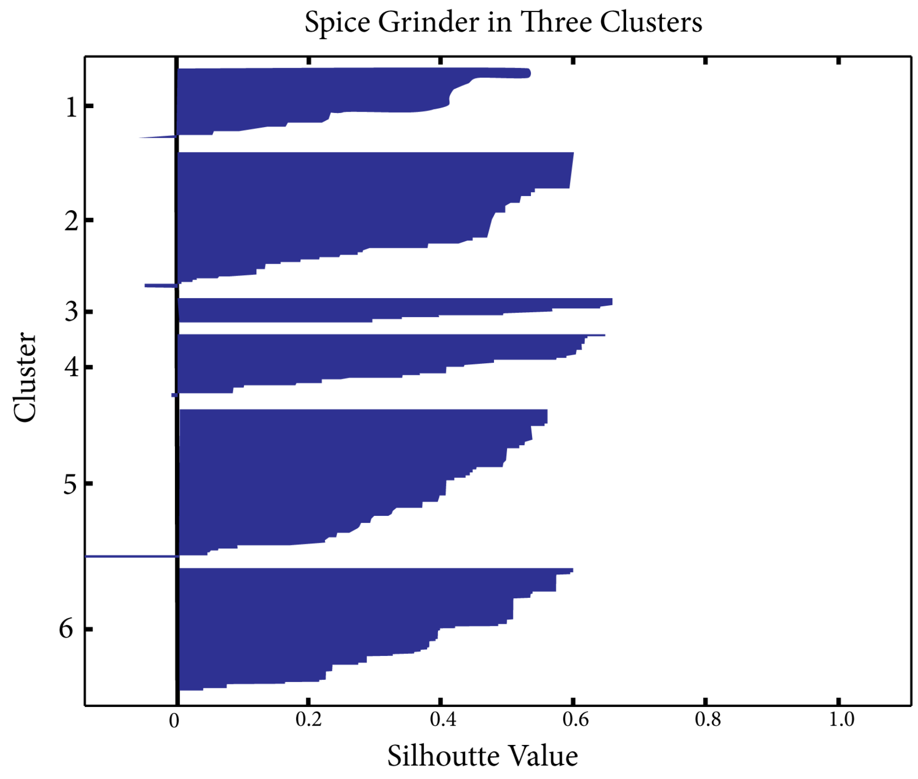


Figure 2.11 Silhouette Plot of Spice Grinder in Six Clusters

Four concepts were identified as outliers that did not appear to belong to any of the six clusters, as indicated by the negative values on the plot. On further inspection these concepts were ill formed, and did not appear to represent workable solutions. The results of this problem are the least useful, from a user standpoint, of the three. Though concepts were grouped into six general types of solutions, further inspection showed that the six types of solutions were very similar. All solutions were essentially a spinning blade and some kind of container; the different groups represented differing arrangements of the same basic sets of components.

2.3.4 Conclusions on Component Based Sorting Through Clustering and PCA

These test problems provide preliminary evidence that Principal Component Analysis can be used to effectively simplify the output of automated concept generators. These simplified representations can then be used to efficiently cluster similar concepts using partitional clustering techniques such as K-Means. This clustering organizes the large number of undifferentiated results into a small number of groups that can be more easily understood an evaluated by a human designer.

The results support the assertion that the thousands of results produced represent permutations of only a few basic solution types. Based on these results we can conclude that variable reduction followed by clustering is one possible means to improve the utility of automated concept generators.

A significant drawback to the proposed approach is suggested by the third sample problem. The method treats all function to component transformations as equivalent. For example, converting chemical energy to mechanical energy with an internal combustion engine is just as important as coupling two solids with bolt. If an automobile is the intended result of our design process, different approaches to energy conversion are much more interesting than exploring a variety of fasteners. This is a significant drawback to this approach, as is the need to generate a large sample of concepts, many of which will later be rejected.

2.4 Conclusions

Chapter 2, discussed initial steps on the road to improving the automated concept generation user experience.

First, results were sorted based on predicted concept parameters. While this approach yielded an estimation technique that has proven valuable in other work, its value to the concept generation process his greatly diminished by the high overhead associated with its application.

Patterns observed in concepts sorted through estimated parameters led to the hypothesis that component based sorting could provide the same result but with much less front-end work for the designer.

Practical application of a component based sorting scheme showed the need for variable reduction and automated clustering. While initial results were promising drawback remain. The approach requires a large sample of concepts be generated and then sorted.

The results of clustering are not always meaningful because the algorithm can group concepts based on variations that are inappropriate or uninteresting given the problem at hand.

As suggested in Chapter 1, an ideal approach to concept generation would produce just the most interesting members of the set of all possible results. This chapter, in passing, noted the use of vectors to represent concepts and variable reduction through matrix transformation. These tools can be used in concert with data in a design repository and computational techniques from other disciplines to arrive at that result. Ultimately success depends on the ability of the computer to recognize salient differences in products and concepts. Chapter 3 surveys concepts from design by analogy which lead to a better understanding of how concept similarity can be assessed and provide a benchmark for new techniques developed in later chapters.

3 Functional Analogy in Design

3.1 Introduction

The following chapter deals with measuring functional analogy in design. Functional analogy is not a primary thrust of this work; instead, the goal is to develop a universal approach to measuring similarity in vector space representations of products in early design. Functional analogy measurement is explored because it operates on a specific vector space representation, namely a product-function vector and computes distances between these vectors in an n-dimensional space of possible functions. While it is not possible to directly formulate a universal similarity measure from current work in the area, current approaches can provide a useful benchmark for techniques this work will advance.

3.1.1 Roadmap

This chapter begins by exploring the role and importance of analogy in design. A brief summary of the literature is presented. Then, a quantitative measure of functional analogy is identified from the literature for further exploration. The implementation of this measure is explored in detail. It is applied to products in the design repository to measure their functional similarity to one another. Test cases are developed based on an examination of products in the repository to evaluate the measure's results. The results form a benchmark against which to test measurement methods developed later in this work.

3.1.2 Contributions of this Chapter

1. A review of the design literature and identification of a useful function based analytical measure of similarity
2. Application of existing measure to products in a design repository shows interesting relationships that can form the basis of future work
3. Development of four test cases for functional similarity assessment from the design repository
4. Benchmark based on similarity assessments of current method for evaluating tools that will be developed in the following chapters.

3.2 Measures of analogy in design

An analogy is "a thing that is comparable to something else in significant respects [74]." Analogy can be a powerful tool during ideation and problem solving. If the problem to be solved is like a problem that has already been solved in another domain, then the solution to the out of domain problem can likewise be adapted. Otto and Wood identify it has an intuitive technique for idea generation and suggest looking for analogies in nature and in other product domains based on function [75]. Ullman likewise recommends functional analogies for generating concepts, but cautions that they do not necessarily lead to good solutions by citing the example of aviation pioneers who sought to emulate flapping flight in birds [7]. Ullrich and Eppinger also cite analogies as a useful technique for generation solutions, and imply that it is common amongst experienced designers[9]. It could be argued that ARIZ and the contradiction matrix from TRIZ [3], are predominately an analogy engine. A more extensive survey of introductory design texts would certainly turn up more references to design by analogy and analogical thinking in design.

A significant fraction of recent effort in the area has been devoted to analogies for bio-inspired or biomimetic design [76-79]. Beyond biological analogies, researchers have sought to use visual or physical analogies to stimulate ideation [80-82], while others have focused on textual or linguistic analogies [83]. In all analogy based efforts, a key component of the process is a similarity measure [84], lack of such a measure has long been a problem in mechanical design [85]. This is a problem this dissertation seeks to rectify.

It is well documented that experts and novices use analogize in different ways. Adelson found that expert programmers tended to form abstract representations of problems, while novices preferred concrete representations. The former were better able to see connections between problems and domains and performed better in programming tasks as a result [86]. Later work by the same author posited that analogies facilitated learning to program [87]. Hewett and Adelson in a study of design methods of human computer interaction go so far as to suggest that analogical reasoning typifies an engineering approach to problem solving in that field [88]. Linsey et al have studied the effects of representation on analogizing, and have developed techniques to help reformulate a design problem to facilitate retrieval of analogies [83, 89-92]. The focus of this work is not design by analogy, so a complete survey of the field will not be undertaken here. The above works suggest the variety of research in the area and the importance of analogy in design.

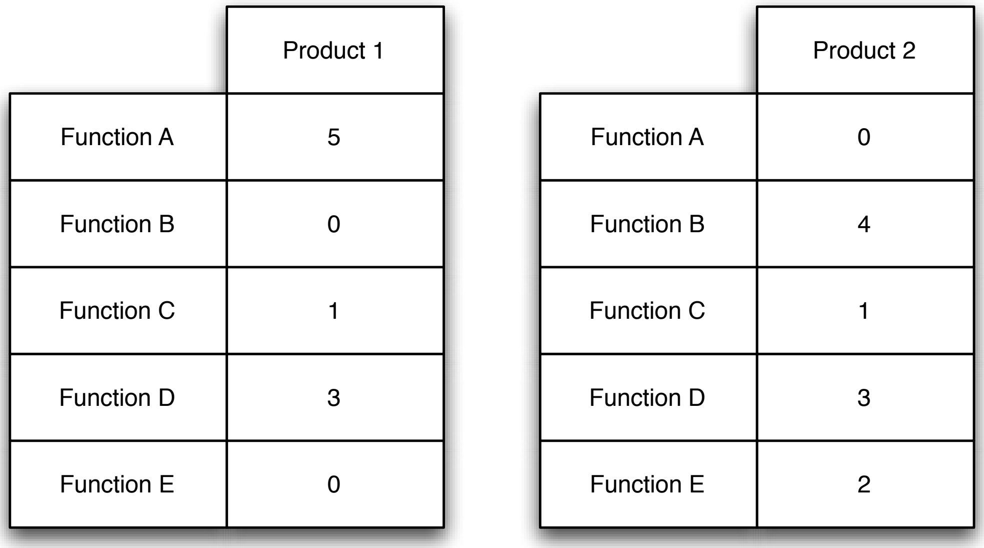
3.3 Application of the Current Measure to Inter- Product Functional Similarity

If analogy implies that two objects or ideas are comparable, it is reasonable to ask how comparable they are. How much like my problem X are already solved problems A,B, and C? It would be convenient to express this measure of analogy or similarity as a quantitative measure. If we are seeking functional analogies in product design amongst a set of existing products, i.e., already solved problems, such a method has been proposed and validated by McAdams and Wood [93, 94]. The following sections will explore this method, and will use it as a benchmark against which to test a more universally applicable vector space similarity measure.

3.3.1 The Method of McAdams and Wood

McAdams and Wood have developed a quantitative analytical measure of functional similarity between products[94]. Their approach is distinct from other work in the field in that it is amenable to a fixed taxonomy of function and flow, and that it bases its comparison on product sub-function. It is an approach that is primarily focused on meeting the needs of engineers engaged in the design of a new artifact. Given a functional description of the product under development, the technique measures the degree of functional analogy or similarity between it and a set of reference products from which analogies are to be drawn. A necessary precursor to their approach is a set of reference products, which have been functionally decomposed following the methods discussed in Chapter 1.

This is a vector space representation of the data; each reference product can be represented as vector whose elements indicate the importance of a particular sub function as assessed by customers. As discussed in Chapter 2, this space can be quite large for the fixed taxonomy of function and flow employed in this work. Figure 3.1 Example Product Function Vectors shows examples of these vectors.



	Product 1
Function A	5
Function B	0
Function C	1
Function D	3
Function E	0

	Product 2
Function A	0
Function B	4
Function C	1
Function D	3
Function E	2

Figure 3.1 Example Product Function Vectors

For this trivial example, we could visually compare dimensions of similarity between our problem, also expressed in this vector space and these product function vectors. However, given that these vectors can be quite large, and that we have access to a large reference set some analytical means of comparison is required.

McAdams and Wood construct a product-function matrix Φ by assembling the vectors of individual reference products. Their method of measuring analogy proceeds by applying two key assumptions. First, all reference products are equally important, or that the only interesting dimension of difference is their relevance to the problem. Second, the number of sub functions in a product is an indication of its complexity, and that this variation in product complexity must be normalized to avoid impact on the similarity assessment. Finally, the measure takes into account customer needs to assess the importance of individual functions. So if in the product to be designed, transforming electrical to mechanical energy is deemed very important based on customer requirements, the method will treat as more similar reference products where this was an important function.

Employing these requirements and assumptions, the following step-by-step process converts a product-function Φ into a normalized product function matrix N suitable for measuring functional similarity.

The elements of N are simply the elements of Φ weighted by the following function.

$$v_{ij} = \phi_{ij}\left(\frac{\bar{\eta}}{\eta_j}\right)\left(\frac{\mu_j}{\bar{\mu}}\right)$$

The average customer rating $\bar{\eta}$ is simply the sum of all elements in Φ divided by the number of columns in Φ. If Φ is an mxn matrix, this can be calculated using the following equation.

$$\bar{\eta} = \frac{1}{n}\sum_{i=1}^{m}\sum_{j=1}^{n}\phi_{ij}$$

While the total customer ratings for each product are merely the column sums of Φ.

$$\eta_j = \sum_{i=1}^{m}\phi_{ij}$$

The number of functions in each product and the average number of functions per product are given in turn by the following equations.

$$\mu_j = \sum_{i=1}^{m}H\left(\phi_{ij}\right)$$

$$\bar{\mu} = \sum_{i=1}^{m} \sum_{j=1}^{n} H\left(\phi_{ij}\right)$$

H is a Heaviside function.

$$H(x) = \begin{Bmatrix} 1 \text{ when x} \neq 0 \\ 0 \text{ otherwise} \end{Bmatrix}$$

Once the weighted product vectors have been calculated, the columns of N are renormalized so that their norm is unity to yield N'. In this form the similarity between the product under design and any reference product is the inner product of the vector space representations of each. The closer this value is to 1, the more similar the two products.

To better illustrate this process, consider the following problem. Imagine the following product-function matrix. Each column represents a reference product, each column a distinct function, and each element an integer representing the perceived importance of each function in each product.

$$\Phi = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 3 & 3 & 6 & 5 \\ 5 & 0 & 4 & 2 \\ 2 & 0 & 7 & 1 \end{pmatrix}$$

Suppose a product must be designed to meet those four functions where the importance of each is shown in vector s.

$$s = \begin{pmatrix} 5 \\ 3 \\ 8 \\ 1 \end{pmatrix}$$

Following the weighting scheme outlined above Φ is transformed to N.

$$N = \begin{pmatrix} 1.2 & 0 & 0 & 1.5 \\ 3.6 & 3.3 & 3.5 & 7.4 \\ 6.1 & 0 & 2.4 & 3.0 \\ 2.4 & 0 & 4.1 & 1.5 \end{pmatrix}$$

Which when normalized becomes approximately:

$$N' = \begin{pmatrix} 0.2 & 0 & 0 & 0.2 \\ 0.5 & 1.0 & 0.6 & 0.9 \\ 0.8 & 0 & 0.4 & 0.4 \\ 0.3 & 0 & 0.7 & 0.2 \end{pmatrix}$$

The inner products of s and the columns of N are then calculated.

$$\Lambda = \begin{pmatrix} 0.9 & 0.3 & 0.6 & 0.7 \end{pmatrix}$$

From this analysis the designer would concluded that the first reference product is most analogous followed closely by the fourth and third products. The second product is a poor match, as would be concluded by inspection.

Application to Data in the Design Repository

The method outlined above, is not a suitable solution to measuring concept similarity in automated concept generation because it was developed to assess functional similarity. All automatically generated concepts should have identical functionality. While the method could be modified, the goal of this dissertation is to develop a universal approach to measuring similarity amongst vector space representations in design. The method proposed by McAdams and Wood is presented in detail here because it can serve as a meaningful benchmark for techniques that will be developed. Toward that end this section will explore the application of this method to the set of products contained in the design repository. This will provided a benchmark of similarity measurements to test proposed techniques against. Similarity measures developed later in this work should perform as well as, or improve upon these benchmark measures.

Equivalence with the existing method is easy to establish, rank orderings of product similarity should agree, or very nearly agree. To establish that a measure improves on the current assessment, we can identify groups of products in the repository that we assert are similar. If the current measure categorizes any of these test cases incorrectly, it may be possible to construct a measure that rectifies this error.

There are, at the time of writing, 167 products in the repository including consumer scale electromechanical products, biological systems, and spacecraft subsystems. Four subsets of products are identified that will serve as test cases. First, there are 4 coffee makers in the repository. These products should be, functionally speaking, more similar to each other than to any other products. Second, there is a set of biological systems that were recorded as part of a bio-inspired design investigation. Due to the unique flow types observed in these systems, they should be more similar to each other than to any electro-mechanical product. Third, there is a set of products related to product failure reports. These, like the biological systems should be self-similar, but relatively different from other artifacts. Finally there are four drills in the repository. As with the coffee makers, it is assumed that these should be more similar to each other than to any other product. Table 3.1 Similarity Metric Test Cases summarizes the identified test cases.

Table 3.1 Similarity Metric Test Cases

Test Case	Description	Products
1	Coffee Makers	Black 12 cup deluxe, black 12 cup economy, black 4 cup regular, white 4 cup economy
2	Biological Systems	Any artifact where system type is biological
3	Aerospace Systems	A set of artifacts representing aerospace systems recorded to capture failure data
4	Drills	Skil drill, firestorm drill, delta drill, b and d drill attachment

Use of the data in the repository presents a problem. At present, no system in the repository has information about customer needs tied to product functions. The method of McAdams and Wood will have to be applied by treating the customer needs derived importance of each function as being equivalent. Obviously, this diminishes some of the utility of their approach, but making this assumption will still allow for useful results.

To assess the functional similarity amongst products in the repository, the following procedure is employed. First, a product function matrix is f retrieved by querying the design repository. The sequel query necessary to retrieve this data is available in Appendix A. The results of the query must be further processed to account for the hierarchy of functions and flows in the Functional Basis. For example, if the database records that a particular product includes the function of *import liquid,* it must also include the equivalent parent function and flow, *channel material.* The generated product function matrix then contains functions as entered, and at any appropriate higher levels of generality. This compensates for the varying level of detail at which products are recorded.

The method presented above is then applied directly to the data. There are many questions that can be asked of these data. First, what if any general trends can be observed in the inter-product functional similarity measures? The following plot Figure 3.2 shows a plot of the ordered measures of similarity between products. Each line represents an individual product with each point showing the measured similarity to the nth most similar product.

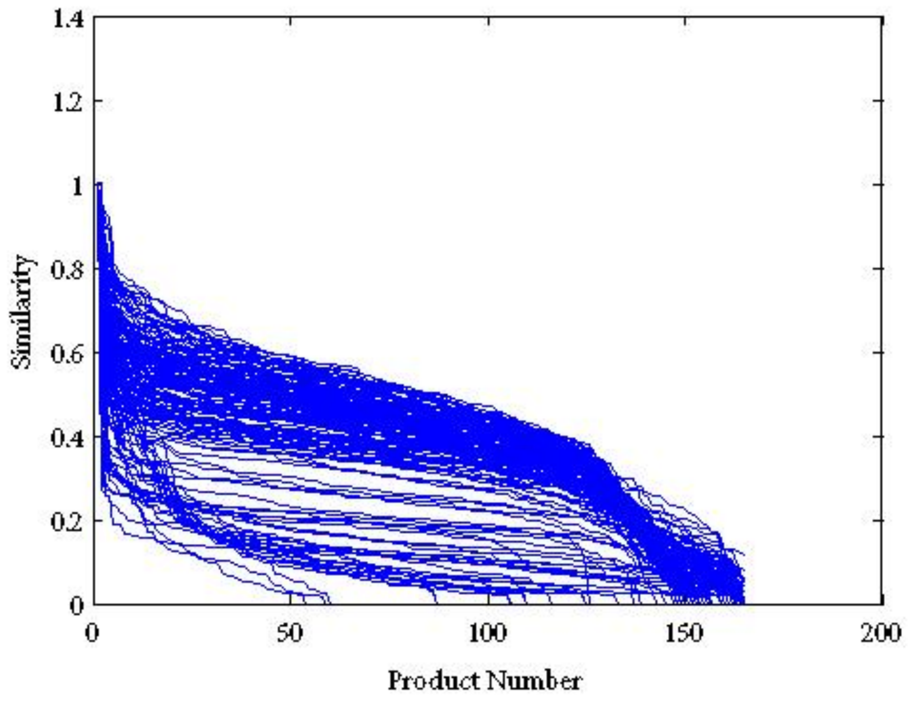


Figure 3.2 Plot of Product-Product Similarity Measures

An examination of this plot shows that there are, in general, three types of products. The first is very similar to a few products, but then its degree of similarity drops precipitously and the majority of other products in the repository are dissimilar. The second also has strong similarity to a few products, but the remainder of the set is at least weakly similar. The majority of products fall into this category. Finally a third group has a weak but relatively homogenous degree of similarity with all products in the repository. This is a relatively interesting group; its members are products that are relatively unique amongst the set of products in the repository, so it merits some additional study. The following plot Figure 3.3 Histogram of Distance of Most Similar Product shows a histogram of the magnitude of the similarity measure of the product most similar to each product in the repository.

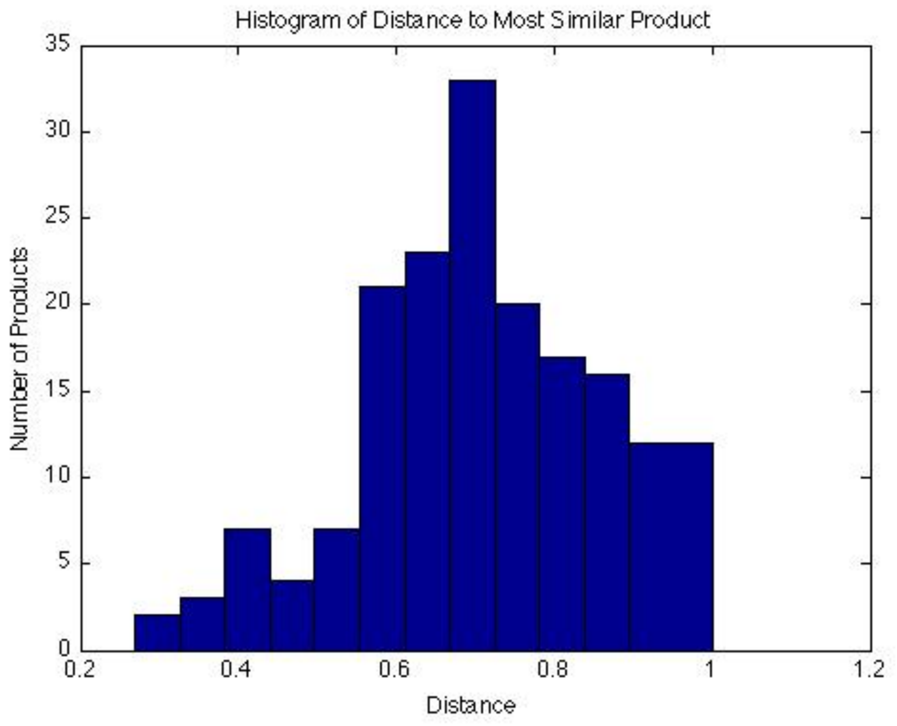


Figure 3.3 Histogram of Distance of Most Similar Product

The histogram shows that there is a meaningful set of products for which the most closely related product is not very similar. It is helpful to think of the distance measure as a cosine distance, thus a value below 0.5 suggests that the two products are not very similar at all. Table 3.2, presents a summary of these products.

Table 3.2 Products with Low Similarity to Other Members of The Repository

Product	**Distance to Most Similar Product**
dryer	0.54
component basis reference artifacts	0.54
dishwasher	0.54
asm volume 2	0.53
cotton candy machine	0.51
nasa anomaly	0.50
hulk hands	0.50
alcohawk digital alcohol detector	0.49
coolit drink cooler	0.48
ub roller coaster	0.47
turtle	0.45
bolting	0.43
natural sensing	0.42
fly	0.42
lawn mower	0.42
dna	0.41
jar opener	0.40
john deere tractor gear	0.38
brake system	0.37
bat	0.36
two component regulatory system	0.35
lichen	0.30
heart	0.27

Many members of the table are biological systems or products that were originally part of a failure dataset. Those electro-mechanical products which are included in the table may represent unique functions or flows. They may be unique or innovative, or the may simply be from categories of products that have not been sampled extensively in the repository. While this points toward an interesting set of questions, answering them is beyond the scope of the current work.

Instead, we will return to the four test cases identified above and assess how well the current measure meets our expectations of product functional similarity. First, the coffee makers: there are four coffee makers in the repository. It is expected that they be more similar to one another than to any other products. Applying the measure of McAdams and Wood, we find that this is the case; the results for the four most similar products are presented in Table 3.3 Coffee Maker Similarity as Measured by McAdams and Wood Method.

Table 3.3 Coffee Maker Similarity as Measured by McAdams and Wood Method

Reference Product	**Degree of Similarity**			
	1st	**2nd**	**3rd**	**4th**
black 12 cup deluxe	white 12 cup regular	black 4 cup regular	white 4 cup economy	black 12 cup economy
black 12 cup economy	white 12 cup regular	black 4 cup regular	white 4 cup economy	black 12 cup deluxe
black 4 cup regular	white 12 cup regular	white 4 cup economy	black 12 cup economy	black 12 cup deluxe
white 12 cup regular	black 4 cup regular	white 4 cup economy	black 12 cup economy	black 12 cup deluxe
white 4 cup economy	white 12 cup regular	black 4 cup regular	black 12 cup economy	black 12 cup deluxe

The method correctly groups the coffeemakers together, and it finds that numerically they are more similar to one another than to any other product. The measured similarities range form almost one to a low 0.77 in each case the fifth most similar product is significant less similar with values one the order of 0.5. This suggests that the existing method handles the first test case well.

Next, lets examine the drills. There are four drills in the repository, as with the coffee makers we expect them to be more similar to one another than any other products. The following table, Table 3.4, summarizes the results.

Table 3.4 Drill Similarity as Measured by McAdams and Wood Method

Reference Product	**Degree of Similarity**			
	1st	**2nd**	**3rd**	**4th**
b and d drill attachment	b and d sander attachment	b and d mini router attachment	b and d jigsaw	firestorm circular saw
delta drill	delta jigsaw	delta sander	firestorm saber saw	b and d screwdriver
firestorm drill	delta circular saw	firestorm saber saw	delta drill	delta nail gun
skil drill	skil flashlight	skil jigsaw	b and d power pack	b and d screwdriver

Unlike the coffee makers, the drills are, according to the assumption made earlier, entirely miscategorized. An examination of the calculated similarity metrics for each product shows they range from 0.8-0.7, so the identified products are, according to the measure, very similar. Either there is a problem with the measure or with the assumption that the drills are similar. I believe that the drills are, in fact, a poor test case. While they appear superficially similar their actual functionality is quite different. The b and d drill attachment is a drill head for a modular tool system while the other drills are stand-alone products. The delta and firestorm drills are members of

product families, and so it's not entirely surprising that they appear more similar to other members of their respective families. Finally, the drills are frequently categorized as being similar to other power tools, which remove material like saws, so perhaps in this regard the measure is going a good job. For each entry in the table another drill is the 8th, 7th, 3rd,and 25th most similar product, respectively. For the first three, the similarity metric remains between 0.75-0.65. What is so different about the skill drill? An examination of the product shows that the Skil drill was a complex product with a great deal of functionality above what is seen in the other drills. The added functionality makes it similar to a large number of products. In fact the first drill on the list, while the 25th most similar product, still has a similarity measure above 0.6. This analysis suggests that while the rank ordering presented by McAdams and Wood does not precisely conform to preconceptions about similarity it does a good job of identifying products that are truly functionally similar. Even though I've argued that the drills, while superficial similar, are in fact different, I will retain this as a test case because of the interesting results found.
Next, the test case of biological systems is considered. There are currently 32 systems in the repository that are recorded as biological artifacts, or strategies. Amongst this set there are several biological systems that are measured to be most similar to electromechanical products. A closer examination of these shows that in this set no element has a similarity measure to an electromechanical product significantly greater than 0.5. By taking 0.5 as a cutoff for similarity, as discussed above, it can be concluded that the existing method correctly handles all biological systems in the repository. Alternatively, we might conclude that this points to a deficiency in the current measure. These systems are recorded using the same function and flow taxonomy as electromechanically products, so it stands to reason that

functionally some similarity is to be expected. This will be explored further by applying a new candidate similarity measure in Chapter 4.

Finally, there are the products recorded from failure data sets. There are five such systems that recorded data from ASM failure reports, Consumer Product Safety Commission Reports, NASA Failure reports and NTSB rotorcraft failure reports. These products should, in general be very different from the consumer scale electromechanical products found elsewhere in the repository, so they should have a higher degree of similarity with one another than with other systems.

Table 3.5 Similarity of Aerospace Failure Data as Measured by McAdams and Wood Method

Reference Product	Degree of Similarity			
	1st	2nd	3rd	4th
nasa anomaly	garage door opener genie	star scanner	-	-
galileo	rotorcraft	asm volume 1	-	-
cpsc failures	datsun truck	Component basis	-	-
asm volume 1	rotorcraft	galileo	-	-
asm volume 2	gse solar power module 60 watt	cordless kettle	-	-
rotorcraft	asm volume 1	galileo	-	-

Table 3.5 shows the results of the comparison for the set of failure data products. If we apply as distance of 0.5 as a cutoff only the first two most similar products remain relevant. From this set we see mixed results. All products except asm volume two include another failure product, but it typically is not the most similar product.

3.4 Conclusion

Finding analogies can be a useful aid to ideation in design. The preceding sections have discussed the role analogy in design and some results from the design by analogy literature. While no universal quantitative measure of analogical distance was identified, an approach was found in the literature that meets many of the needs identified earlier in this work in the specific area of function based design by analogy. The calculations necessary to construct this measure were discussed in detail. Then the measure was applied to explore inter-product similarity amongst products in the design repository. Four test cases were identified and explored. For the biological system and coffee maker test cases, we find that the measure performs as expected. For the failure product and drill test cases, the results contradict initial expectations, but further examination suggests that the expectation may have been incorrect. Interesting behavior was still observed in these two cases, so they are retained for use in evaluating the measures developed in subsequent chapters. The justification for retaining them is that in these products is a set that confounds expectations of functional similarity. These product sets seem like they should be similar, but applying the current similarity measure suggests plausible reasons why they are not. If a new measure can plausibly sort these in a way that matches expectations, that would be an interesting result. These test cases and the final rank ordered measures of product-to-product similarity form the benchmark against which future work will be evaluated.

4 A Universal Approach to Vector Space Similarity Measurement in Engineering Design

4.1 Introduction

The following chapter deals with identifying and validating an approach to vector space similarity measurement that, unlike the method discussed in the prior chapter, is amenable to a wide variety of computational design tasks. The results of the previous chapter form the basis for comparison and analysis of the new method. This chapter is one of the primary contributions of this dissertation. Identifying and verifying a method of vector space similarity measurement that is not task specific can contribute to a variety of design methods and tools, particularly analogical design tools. In addition, the particular method explored in this chapter provides a bridge between engineering design and information retrieval, a field with a rich body of tools and methods that may be of use for a variety of problems in design.

4.1.1 Chapter Roadmap

This work has used and relied on vector space representations of design data without explicitly defining what a vector space model is, so the chapter begins with a more rigorous definition of vector space models. After arguing that many of the data sets and product representations engineering design research seek to manipulate are vector space models, specific results and techniques from the field of information retrieval are introduced. A parallel between the similarity measures needed in this work and the approach used to find similar texts in a corpus with latent semantic indexing is drawn. Recognizing that the problems are analogous, the solution presented by latent semantic indexing is adapted to suit the vector space models for engineering design. The new similarity measure is compared to the measure of functional analogy explored in chapter 3. Comparing and contrasting the new measure with the old provides preliminary evidence to support its utility for engineering design problems. Finally, the modifications necessary to the new method to provide a replacement for the tool discussed in Chapter 3 are presented.

4.1.2 Contributions of this chapter

1. Verifies that important representations of engineering design data meet the formal definition of a vector space model
2. Adapts and extends latent semantic indexing, an information retrieval technique to formulate a new method for assessing similarity in vector space engineering design data.
3. Applies the resulting similarity measure to the functional analogy problems of previous chapter, and shows that the new measure meets or exceeds the performance of the existing methods for the benchmark problems
4. Identifies the steps needed to use the new similarity measure in a viable function-based analogy search tool that can replace prior work.

4.2 Vector Space Representations

Within computational design tools, vector space representations are common. Before delving into their applications in engineering design an obvious question is, what are vector spaces? A simple definition of a vector space is any collection of vectors where linear combinations make sense. More specifically a vector space over the real numbers $\mathbb{R}$ consists of a set V and the addition and multiplication operators subject to 10 conditions.

Sufficient conditions for the set V to be a vector space are:

1. If $\vec{a},\vec{b} \in V$ then $\vec{a}+\vec{b} \in V$

2. If $\vec{a}, \vec{b}, \vec{c} \in V$ then $\vec{a}+\vec{b} = \vec{b} + \vec{a}$
3. $(\vec{a}+\vec{b}) + \vec{c} = \vec{a} + (\vec{b} + \vec{c})$
4. $\vec{0} \in V$ and $\vec{0} + \vec{a} = \vec{a}$ for any $\vec{a} \in V$
5. For any $\vec{a}$, -$\vec{a}$ exists and $\vec{a} + (-\vec{a}) = \vec{0}$
6. For scalars g and h in $\mathbb{R}$, $g \cdot \vec{b} \in V$
7. $(g + h) \cdot \vec{b} = g \cdot \vec{b} + h \cdot \vec{b}$
8. $(\vec{b} + \vec{c}) \cdot g = g \cdot \vec{b} + g \cdot \vec{c}$
9. $(g \cdot h) \cdot \vec{b} = g \cdot (h \cdot \vec{b})$
10. $1 \cdot \vec{b} = \vec{b}$

These definitions and conditions are adapted from[95] and [96]. While it is typical to imagine vector spaces as columns or rows of real numbers, a vector space is more correctly described as any collection where these linear combinations work. Fortunately, in this work vector space representations will generally be columns in $\mathbb{R}^n$, and most commonly in $\mathbb{Z}^n$, the set of integers.

A common product representation in engineering design is a Design Structure Matrix (DSM). The DSM is a matrix that describes connections and dependencies among the sub-assemblies and components that make up a product. There are a number of DSM variants today, but most are derived from Steward's design structure system [97]. Originally an attempt to manage the iteration inherent in designing systems with complex interactions, DSMs have evolved to serve a number of different requirements and as a result a number of sub-types of DSM are now recognized[98].

One variant of the DSM is a component or product architecture type. This is a matrix whose rows and columns represent components and whose elements represent connections between those components. For example, if component i connects to component j then the ij^{th} element of the DSM is a count of the number of times that connection is seen in the data. If they do not connect then the ij^{th} element is 0. Such a DSM could be constructed to show the interconnections of an individual product, or component-to-component relationships in a dataset of many products.

It is relatively easy to show that a DSM constructed in this way is a vector space. To validate this claim, consider two products: a vegetable peeler and an ink pen. DSMs for both products are shown below.

	Blade	Handle	Blade Cover
Blade	1	1	1
Handle	1	1	0
Blade Cover	1	0	1

Figure 4.1 Vegetable Peeler DSM

	Barrel	Cartridge	Cap
Barrel	1	1	1
Cartridge	1	1	1
Cap	1	1	1

Figure 4.2 Ink Pen DSM

With these two example DSMs, it can be shown that the vector space conditions outlined earlier in the section are met by DSMs. First, these two matrices are in the set of all components. Their sum, shown below, is also in that set, and addition is both transitive and commutative. An example of addition is shown in Table 4.3.

	Blade	Handle	Blade Cover	Barrel	Cartridge	Cap
Blade	1	1	1	0	0	0
Handle	1	1	0	0	0	0
Blade Cover	1	0	1	0	0	0
Barrel	0	0	0	1	1	1
Cartridge	0	0	0	1	1	1
Cap	0	0	0	1	1	1

Figure 4.3 Ink Pen and Vegetable Peeler Combined DSM

It is useful that DSMs and other design representations exhibit this property of vector space representations because it enables the construction of matrices that incorporate knowledge about a body of products. A variety of efforts have taken advantage of this property of DSM's and other matrices in engineering design. Aggregated vector space models of product functions and component selection have been used as an aid in concept generation[99, 100], and extended to build a functional automated concept generator [20]. Various related efforts have explored predicting product failure [55, 57, 60], assessing manufacturability [62, 101], and estimating environmental impact in early design from similar datasets [66].

Returning to the question of whether the DSM and by extension analogous representations are proper vector spaces, criteria four through ten must still evaluated. The next three criteria can be checked by inspection. Clearly addition of a zero vector will not change the result. Likewise, inspection shows that the addition of the negative of one of these DSMs to itself would result in an empty DSM. Finally, the multiplication of any of these by a scalar will scale the data but it will still remain in the set.

The two example products used share no components in common. They are orthogonal to one another, so their product should be a zero matrix. They will not suffice to demonstrate any of the multiplicative criteria. An alternative pen design without a cap is introduced in the following DSM to facilitate examples of multiplication.

	Barrel	Cartridge	Cap
Barrel	1	1	0
Cartridge	1	1	0
Cap	0	0	0

Figure 4.4 Alternative Ink Pen DSM

Let the first and second pens be $\vec{b}$ and $\vec{c}$ and let g and h be any scalars. Then criteria 7 through 9 are satisfied as shown in the following figures. Figure 4.5 Criterion 7 shows that the sample DSM meets the seventh criteria; scalar multiplication is distributive with respect to vector addition.

$$(g+h)\cdot\vec{b} = g\cdot\vec{b} + h\cdot\vec{b}$$

$$(g+h)\cdot\begin{pmatrix}1&1&1\\1&1&1\\1&1&1\end{pmatrix} = g\cdot\begin{pmatrix}1&1&1\\1&1&1\\1&1&1\end{pmatrix} + h\cdot\begin{pmatrix}1&1&1\\1&1&1\\1&1&1\end{pmatrix}$$

$$\begin{pmatrix}g+h&g+h&g+h\\g+h&g+h&g+h\\g+h&g+h&g+h\end{pmatrix} = \begin{pmatrix}g&g&g\\g&g&g\\g&g&g\end{pmatrix} + \begin{pmatrix}h&h&h\\h&h&h\\h&h&h\end{pmatrix}$$

$$\begin{pmatrix}g+h&g+h&g+h\\g+h&g+h&g+h\\g+h&g+h&g+h\end{pmatrix} = \begin{pmatrix}g+h&g+h&g+h\\g+h&g+h&g+h\\g+h&g+h&g+h\end{pmatrix}$$

Figure 4.5 Criterion 7 Distributivity of Scalar Multiplication

Figure 4.6 demonstrates the eighth criterion, distributivity of vector sums with scalar multiplication.

$$(\vec{b}+\vec{c})\cdot g = g\cdot\vec{b}+g\cdot\vec{c}\left(\begin{pmatrix}1&1&1\\1&1&1\\1&1&1\end{pmatrix}+\begin{pmatrix}1&1&0\\1&1&0\\0&0&0\end{pmatrix}\right)\cdot g = g\cdot \begin{pmatrix}1&1&1\\1&1&1\\1&1&1\end{pmatrix}+g\cdot\begin{pmatrix}1&1&0\\1&1&0\\0&0&0\end{pmatrix}\left(\begin{pmatrix}2&2&1\\2&2&1\\1&1&1\end{pmatrix}\right)\cdot g = \begin{pmatrix}g&g&g\\g&g&g\\g&g&g\end{pmatrix}+\begin{pmatrix}g&g&0\\g&g&0\\0&0&0\end{pmatrix}$$

$$\begin{pmatrix}2g&2g&g\\2g&2g&g\\g&g&g\end{pmatrix}=\begin{pmatrix}2g&2g&g\\2g&2g&g\\g&g&g\end{pmatrix}$$

Figure 4.6 Criterion 8 Distributivity of Vector Addition with Scalar Multiplication

Figure 4.7 Criterion 9 verifies the associative property of scalar multiplication.

$$(g\cdot h)\cdot\vec{b} = g\cdot(h\cdot\vec{b})$$

$$(g\cdot h)\cdot\begin{pmatrix}1&1&1\\1&1&1\\1&1&1\end{pmatrix}=g\cdot\left(h\cdot\begin{pmatrix}1&1&1\\1&1&1\\1&1&1\end{pmatrix}\right)$$

$$(gh)\cdot\begin{pmatrix}1&1&1\\1&1&1\\1&1&1\end{pmatrix}=g\cdot\left(\begin{pmatrix}h&h&h\\h&h&h\\h&h&h\end{pmatrix}\right)$$

$$\begin{pmatrix}gh&gh&gh\\gh&gh&gh\\gh&gh&gh\end{pmatrix}=\begin{pmatrix}gh&gh&gh\\gh&gh&gh\\gh&gh&gh\end{pmatrix}$$

Figure 4.7 Criterion 9 Associative Property of Scalar Multiplication

Finally, by inspection we can see that the product of the DSM and one is one, so criterion 10 is satisfied.

The proceeding discussion illustrates that DSM's are vector spaces. A similar approach could be used to show that many other matrix and vector data structures employed in the engineering design literature are also vector spaces. Widely accepted representations like the morphological matrix as described by Pahl and Beitz [5] and the design matrices of Suh [1, 2] meet the definition of a vector space. Even graphs, which aren't obviously vector space models can be transformed into adjacency and incidence lists or matrices that meet all the criteria of a vector space. So many important design representations meet the definition of a vector space that the challenge would be to find a numerical or categorical representation of product data that is not a vector space model. Such a representation, by definition, would not have useful linear combinations, and would likely be of limited utility, particularly in computational design. If we accept that most representations are vector space models, then this leads inevitably to the conclusion that the proper route to resolving the difficulties in similarity measurement highlighted in preceding chapters is to adapt vector space similarity measures. The following sections will explore adapting some of these techniques from other domains.

It should be obvious that a vector space, as defined above, is a flexible model that could reasonably apply to a variety of systems. When referring to vector space models, it's necessary to be more explicit about the level of abstraction being applied. As suggested by Dubin [102], four levels of abstraction can be recognized in a vector space model. First, there are algebraic vector spaces. This is the most general category, and simply requires that the linear algebraic axioms discussed at the beginning of the section hold. These are outlined in standard linear algebra texts [95]. Second, there is the measurement theory view of vector spaces. As described in Michell [103], this view sees the vector space as defining quantitative, ordinal relationships amongst variables such that the distance between observations is a function of the differences between the vector elements. This is one view that we will exploit, particularly in assessing the similarity of automatically generated concepts. A third view is the one that most engineers will be familiar with: the vector space as a model of physical forces and relationships. This might include the speed and direction of the particles of a body in motion [104], or the state of stress inside an object [105]. Finally, there is a data-centric interpretation. In this view data are represented in a matrix where items are represented along one dimension of the matrix while observations about particular features are recorded on the other [106]. Operations can be understood to apply in the same way that they do in the more abstract algebraic sense, but concepts like distance and orthogonality are necessarily less well defined. This is the interpretation that will be used most heavily in this work.

4.3 Vector Space Techniques in Information Retrieval

Setting aside, for the moment, the problem of measuring the similarity of two design artifacts against some arbitrary criteria, consider another area where measuring the similarity of two artifacts is important, information retrieval. The field of information retrieval concerns itself with finding relevant information in a document, article, book, or website based on a particular query. Researchers in the field have necessarily had to confront the problem of assessing the similarity of sets of complex artifacts

4.3.1 History of Vector Space Models in Information Retrieval

The advent of computers, databases, and digital documents has fundamentally alerted the challenges and opportunities in this area. On the one hand the volume of content created on a daily basis is staggering. The barriers to creating and publishing a document, whether in print or electronically, have never been lower. On the other, the same tools that allow for the easy dissemination of information provide opportunities to automatically parse, organization, and locate relevant works.

Traditional information retrieval is accomplished through some form of indexing. In the early years of the written word, that index was often contained solely in the head of the owner or curator of a particular library. Eventually libraries grew too large and users would instead consult physical catalogues to locate works by a particular author, about a particular subject, or with a specified title. The first real changes to this arrangement followed the rise of the personal computer. However, early generations of computational tools simply replaced the physical catalogue with a database containing the same information. The indexing of information remained a mostly human driven process. This quickly became an untenable situation. There are millions of books in print in the United States, and Google indexed its one trillionth unique URL in 2008 [107]. It would be utterly infeasible for a small team of curators or librarians to digest and index all this data. As the number of people indexing the information grows, inconsistency is certain to creep into the data set. Imagine two different indexers reading this dissertation. One might index it using terms like engineering design and computation in design. Another reader might choose different words like mechanical engineering and vector space methods. If enough readers indexed the document, these inconsistencies might cancel each other, but in the near term the random assignment of a particular human indexer to a document could result in important information being hidden from interested readers. A robust and preferably automated indexing scheme is needed.

Driven by the increased availability of networked computers, aggressive development of modern automated information retrieval tools began in the early 1990's with the beginning of the Text Retrieval Conference (TREC) under the auspices of the Defense Advanced Research Projects Agency (DARPA) and the National Institute of Standards and Technology [108]. Meanwhile a related, concurrent DARPA effort called TIPSTER sought to identify a common set of protocols for automated document processing [109]. The problems faced by the information retrieval community are similar to the engineering design problems addressed by this dissertation. In both, the similarity of individual members of a group of objects must be assessed automatically. Singhal identifies three different approaches in the information retrieval literature, vector space models, probabilistic models, and inference networks [110]. The remainder of this chapter will examine vector space methods in information retrieval and apply them to similar problems in engineering design to evaluate their utility.

One avenue explored in information retrieval was to represent documents as a vector of the frequency of term occurrence [111]. The application of a vector space model to document classification and retrieval is typically credited to Salton [112]. However recent research by Dubin, ironically using methods dependent on vector space models, finds a more complicated origin [102]. Switzer [113] and Sammon [114] also explored vector space models at approximately the same time, and there were likely other efforts that have been lost because of the large gap between the theoretical exploration of vector space methods in the 1960s and their practical implementations in the late 1980s. Since it was first described, the application of vector space models to document indexing has been the subject of continuous evolution and development.

Tversky pointed out the limitations of a vector model to mimic human judgments of similarity. The geometric interpretation of similarity in the vector space model does a poor job of dealing with issues of context. For example, a naïve vector space model would struggle to determine if Miami and Havana are similar, or different. They have similar locations and climates, but they're quite different politically and economically. Without appropriate structure the vector space model will miss these nuisances. Humans, unlike the vector space model also respond to different stimuli in different ways. The threshold of similarity for two things to sound alike is different that the threshold for which the look alike [115]. This is an important issue in using these tools in engineering design applications. The choice of a diesel engine instead of a four-cycle engine in a car seems less dramatic than the choice of a jet engine instead of an internal combustion engine for an airplane. Approaches to deal with these context specific issues will be discussed in more detail in later sections on term weighting.

Theophylactou and Lalmas extended the vector space model by combining it with elements from theories of evidence to create a model better equipped to deal with natural language [116, 117]. While Arampatzis et al employed an indexing scheme derived from linguistics as the basis for indexing documents [118], Jiang and Littman approximated high dimensional vector space representations to make them more computationally feasible [119]. Meanwhile Salton and others continued to refine and implement vector space tools [111, 120-125]. The direction in recent vector space information retrieval research most relevant to the goals of this research is an approach called latent semantic indexing.

4.3.2 Latent Semantic Indexing and Rank Reduced Approximations

Two key challenges in information retrieval are synonymy and polysemy. Synonymy is the case where multiple words share exact or overlapping meanings. For example, consider searching for work about the properties of the normal or Gaussian distribution in statistics. Many works will use the term normal distribution, others will describe the same distribution as a Gaussian, and some subset of writings will use both names. How can an automated indexing system determine that these works are all about the same subject? How can it know, or at least make a reasonable guess, that if I search for normal distribution, I should also see documents that discuss Gaussians? Polysemy, words with many meanings, presents a related challenge. If an engineer and a psychologist both search a library for articles related to stress and fatigue, each expects an entirely different set of results. How can an automated information retrieval system detect that one subset of results matches the engineer's view of the word stress, while another set matches the psychologist's?

Initially, synonymy and polysemy do not seem like issues that should effect similarity measurement in vector space models in engineering design. As has been discussed in earlier chapters, most product data can be, or has been, encoded using fixed taxonomies of functions, components, and physical parameters. Properly constructed taxonomies should avoid both assigning the same meaning to multiple terms and using multiple terms to express the same meaning. In practice, the results are not so clean. Currently, there are 46 examples of batteries in design repository. Among this set of artifacts, the functionality of a battery is recorded in twelve different ways [15]. If a component with relatively homogeneous functionality like a battery has this many representations, more complicated artifacts are likely to diverge to an even greater degree. Fixed taxonomies for capturing design information will always walk a tightrope between being flexible enough to capture the breadth of information needed to record an artifact and its design rationale and rigid enough to avoid ambiguity. Even with ideal vocabularies to describe it, recorded design information would always be subject to the fallibility of the recorder. Even if information is recorded perfectly, much of that information is the attempt of an outside party to capture the rational of a designer with whom they have probably never communicated. Consequently, there will always be some finite level of uncertainty about recorded design information. It must be assumed that it contains some errors, and does not perfectly capture reality. We can regard polysemy and synonymy as manifestations of uncertainty that are particular to text documents. The approaches used to deal with this particular type of uncertainty can be extended to deal with the various uncertainties inherent in problems in engineering design.

Latent semantic indexing (LSI) begins with a vector space model of information sources, in the case of information retrieval; this is a set of documents encoded as vectors where each component represents the relative importance of a particular term in representing or constructing that document. The aggregation of these vectors into a term by document matrix creates a vector space representation of a corpus of documents [126]. This approach was pioneered by SMART (System for the Mechanical Analysis and Retrieval of Text) [127].

Latent sematic indexing takes this representation a step further by suggesting that these document vectors are a noisy representation of a hidden, or latent, document vector that encodes it’s actual meaning [128].

Constructing low rank approximations of the term by document matrix can reduce the effect of this noise on the query process [129]. Rank reduced representations are used in this context in a variety of applications including mathematics [130], speech recognition [131], modeling noise in structures [132, 133], computer security [134], and image processing [135-137]. This approach has achieved good results for the TREC collections discussed earlier [138], but is most successful when applied to focused collections where, in general, documents are about related subjects [129]. There are a variety of ways to construct rank reduced representations of a matrix including, QR factorization, Principal Component Analysis, Singular Value Decomposition, and Non-negative Matrix Factorization. Once documents have been represented in this rank reduced space, their similarity to one another, or to a query vector can be measured through a variety of methods including Euclidean, Minokowsi, city block, Mahalanobis, or cosine distance, among others.

4.3.3 Constructing the Vector Space Model

This section will illustrate the rank reduction and query matching process as employed in LSI. The following summary will explore the approach and mathematics behind it in sufficient detail to explain the analysis presented later in this section, however LSI is an umbrella term for a complex and still evolving set of techniques more complete surveys are presented by Berry [126, 139-141], Letsche [142], and Pauca [143]. An alternate derivation by Story takes a Bayesian approach to the text classification problem that leads back to LSI [144].

To illustrate the approach, let's consider a particular term by document matrix that represents the titles of several books on my shelf. These books are

Advanced Engineering Dynamics (AED), *Analytical Dynamics* (AD), *Fundamentals of Aerodynamics* (FA), *Mechanical Engineering Design* (MED), and *Product Design* (PD). If we parsed these titles we can extract a series of terms: advanced, engineering, dynamics, analytical, fundamentals, aero, mechanical, design, and product. These terms describe, to a limited extent, the contents of the document. A full term by document matrix would show the frequency of all the important words in each document, and would give a much more complete picture of the subject of each work. The corresponding term by document matrix is illustrated in Figure 4.8.

Title	AED	AD	FA	MED	PD
Term					
Advanced	1	0	0	0	0
Aero	0	0	1	0	0
Analytical	0	1	0	0	0
Design	0	0	0	1	1
Dynamics	1	1	1	0	0
Engineering	1	0	0	1	0
Fundamentals	0	0	1	0	0
Mechanical	0	0	0	1	0
Product	0	0	0	0	1

Figure 4.8 Example Term by Document Matrix

4.3.4 Weighting Schemes

The first step in dealing with a term by document matrix is to apply local, within document, weights and global, across all documents, weights to compensate for the effect of things like document length on term frequency.

Each element of the weighted term by document matrix a_{ij} is equal to the product of a local weight l_{ij}, a global weight g_i, and the original frequency f_{ij}.

$$a_{ij} = g_i l_{ij} f_{ij}$$

A variety of weighting schemes have been proposed, but two popular techniques are inverse frequency weighting, and log-norm weighting [145, 146]. In the inverse document weighting scheme the local weight is 1, while the global weight is given by the following formula [145].

$$g_i = \log_2\left(\frac{n}{df_i} + 1\right)$$

The document frequency df_i is the number of documents in the corpus that contain term i and n is the total number of documents in the corpus. Taking the base two logarithm of the frequency converts the weighting to bits, a common practice in information theory.

Log-entropy weighting involves a more complex set of local and global weights inspired by some results from information theory. Log-entropy weighting begins with the Shannon information content which is traditionally defined by the following equation [147].

$$h(x) \equiv \left\{\begin{matrix} \log_2\frac{1}{P(x)} & 0 < P(x) \leq 1 \\ 0 & P(x) = 0 \end{matrix}\right\}$$

The purpose is to turn the probability of an event into a measure of the information conveyed if we know the outcome of the event. When the probability is zero or one, the outcome is certain and knowing that a particular outcome happened tells us nothing; the information content of the outcome is zero. Now, imagine a box filled with 10 colored balls. Imagine five are orange, four are black, and 1 is white. So the respective probabilities of pulling out an orange, black or white ball are 0.5, 0.4, and 0.1. The information contents of each event are 1, 1.3, and 3.3 bits respectively. If we draw the white ball, we get a lot of information. We know precisely which ball has been drawn and we know that a subsequent draw without replacement will net an orange or black ball. Conversely, selecting an orange ball tells us much less. We know the ball we've selected is one of 5 possible orange balls, and if we draw again the next ball could be any of the three colors. This measure of information content has applications from data compression to optimal measurement schemes, but for the purpose of this work, it suggests how interesting any particular product parameter in our vector space model is.

Table 4.1 Example of Entropy Calculation

		p(x)	h(x)
Orange	5	0.5	1.0
Black	4	0.4	1.3
White	1	0.1	3.3
Total	10	H(x)	1.4

The entropy of an ensemble, or collection of events, is then defined as the average Shannon information content of each outcome.

$$H(x) \equiv \sum P(x)\, h(x)$$

So for the example above the entropy is 1.4 bits. Entropy, from a thermodynamic prospective is the amount of energy unavailable to do work in a heat engine, or the amount of energy required to move heat against a temperature gradient. At the level of particles, this becomes a measure of the order of the particles in the system. Energy is needed to put particles back in a more ordered, colder state, or is lost, as the particles in the system become more disordered with a rise in temperature. The concept of entropy as a measure of order is what gives information entropy its name. Consider two probability distributions, a uniform distribution and a normal distribution. The uniform distribution has much higher entropy because significantly more effort, or sampling, would be needed to understand the structure of the data. In fact, a uniformly distributed random variable has the highest possible information entropy. Local and global weightings under this scheme are then defined by the following equations, where *tf* is the term frequency, *p* is the term frequency divided by total occurrence of the term, and n is the total number of documents in the set [145]. Section 4.4 will explore the effects of different weighting schemes on real product related vector space models.

$$L(i,j) = \log_2(tf(i,j) + 1)$$

$$G(i) = 1 - \sum_{j=1}^{m} \frac{p_{ij}\log_2(p_{ij})}{\log_2(n)}$$

4.3.5 Reduced Rank Approximations

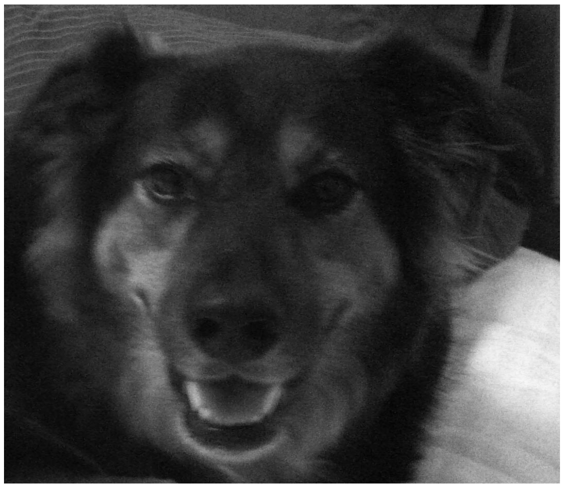


Figure 4.9 Sample Grey Scale Image at Rank 940

Once an appropriate weighting scheme has been applied, the next step is to construct a rank reduced representation of the vector space. The rank of the vector space model is the maximum number of linearly independent vectors [95, 96]. Rank reduced approximations can be thought of as a form of lossy compression. As example, consider a gray scale image like the one in Figure 4.9. The rank of this image is approximately 940. It has around 940 columns representing strips of pixels. Each element of the vector is an integer indicating the value of that pixel on a scale from white to black. This image is represented by a series of numeric vectors that are, conceptually, no different than the data matrices that may be used by various engineering design tools. To illustrate the effect of rank reduction graphically, a series of rank reduced approximations are presented Figure 4.11 and Figure 4.12. The actual mechanism for constructing low rank approximations will be discussed in more detail later in this sections, but for now, this exercise demonstrates the effect of these approximations on data that is readily visualized.

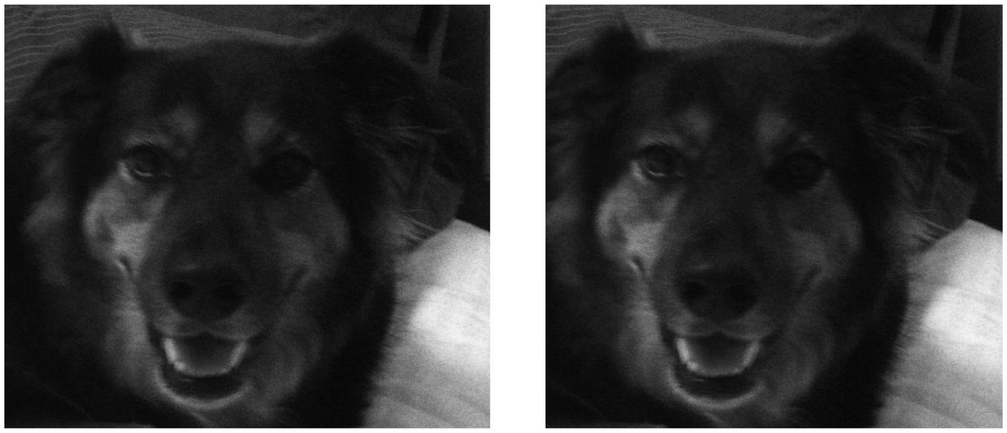


Figure 4.10 Approximation of Image Reduced to Rank 470 and Rank 235

The original image is a relatively clear image of a dog’s face. The next figure shows the image constructed from rank 470 and rank 235 approximations of the original, or roughly half and one quarter the original rank.

In the approximated images, it is still possible to clearly make out the face of the dog, but some changes are evident in the background and in the lighting. As the approximation gets bigger it looks like some detail is lost. Clearly this is a lossy form of compression, not a good way to reduce the size of an image as is clearly shown by much more aggressive approximations of rank 94 and rank 10, or reductions in rank of one and two orders of magnitude.

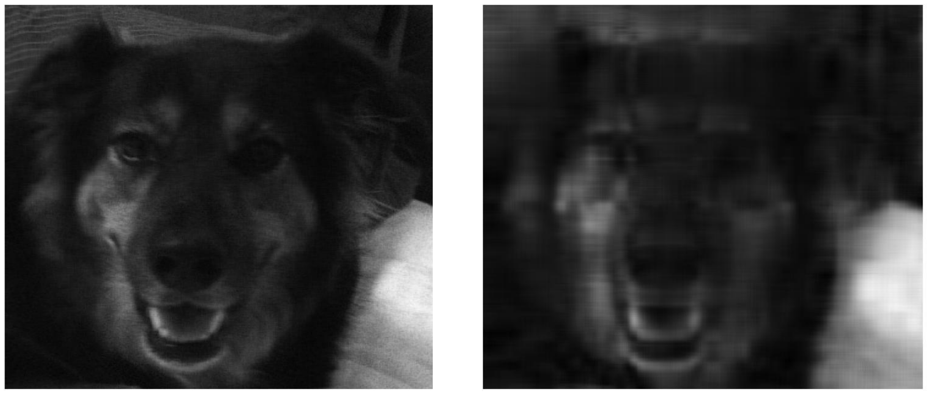


Figure 4.11 Rank 94 and Rank 10 Approximation of Image

The rank 94 approximation still does an acceptable job of representing the original image. Rendered larger, it begins to look fuzzy and detail is clearly lost, but in this case it's possible to construct a rank 94 approximation of the original image that is a reasonable stand in for the original while requiring much less information to construct. The rank 10 approximation on the other hand has clearly gone too far. Without having seen the original it would be difficult to know what the original picture was.

This exercise shows what it means for rank reduced approximation to be a form of lossy compression. It's less clear why this is an appropriate treatment for vector space data being compared for document retrieval in the case of LSI or various applications in engineering design. After all, it seems undesirable to take our data and make it fuzzier. The difference is that an image stores a relatively precise representation of the information we intended to capture. A clear image has been made fuzzy. In LSI, or the engineering applications proposed in this work, the starting point is not a clear image. In LSI the inherent semantic content of the document is essentially hidden and is only viewed through the lens of the specific words used to construct the document. The first chapter of this dissertation attempted to show that the data available to us for many computational design tasks is also a fuzzy and incomplete picture. Consider then, the reduced rank approximation of a noisy image. The following figure shows the original example image, but with Gaussian noise applied. The image next to it is a rank 100 approximation of the noisy original.

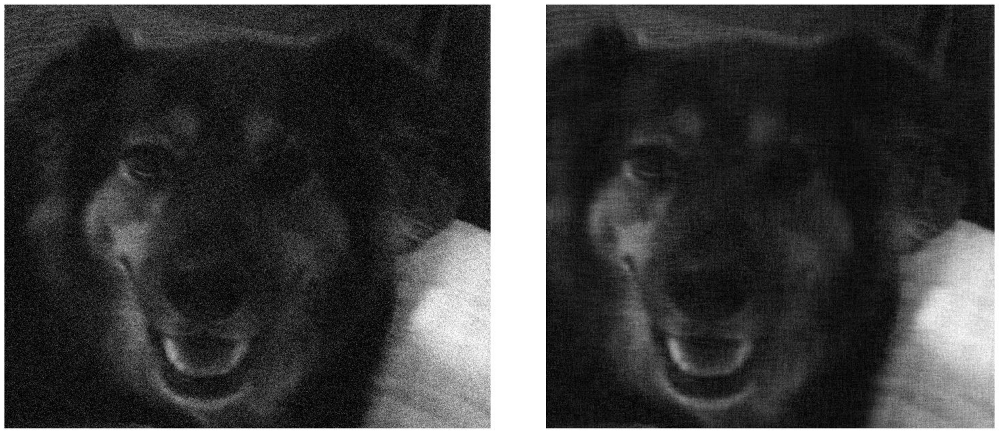


Figure 4.12 A Noisy Image and its Rank Reduced Equivalent

The picture on the left is noisy and grainy. The rank reduced approximation on the right has had the noise smoothed out by the rank reduction process. Reduced rank approximation is, in general not an efficient way to de-noise an image, but researchers in information retrieval have shown that it is a suitable way to deal with the noise and uncertainty in their vector space models. In section 4.4, the suitability of various reduced rank approximations for engineering design data will be assesse, while the remainder of this section will discuss how to calculate and utilize these approximations.

The preceding example gives a qualitative feel for the effect of reduced rank approximations of vector space data; now the actual mathematics can be addressed. Consider the following 3x3 matrix as a motivating example.

$$A = \begin{bmatrix} 9 & 1 & 2 \\ 3 & 8 & 5 \\ 2 & 1 & 9 \end{bmatrix}$$

The matrix is square, positive definite, non-symmetric, and non-sparse. While that makes an initial explanation easier, none of these properties are necessary, and they'll be relaxed as the approach is developed. In fact it should be expected that for realistic vector space models of design artifacts the representations will be sparse, non-square, and symmetric, for certain kinds of data. It's well known that for a given matrix we can find a set of characteristic values, or eigenvalues, and a corresponding set of vectors so that for any eigenvalue λ and any eigenvector $\bar{v}$ such that $A\bar{v} = \lambda I\bar{v}$. Using our example matrix A, we can find a set of eigenvalues by solving the characteristic equation.

$$(A - \lambda I)\bar{v} = 0$$

$$\begin{pmatrix} 9-\lambda & 1 & 2 \\ 3 & 8-\lambda & 5 \\ 2 & 1 & 9-\lambda \end{pmatrix} \bar{v} = 0$$

$$\begin{vmatrix} 9-\lambda & 1 & 2 \\ 3 & 8-\lambda & 5 \\ 2 & 1 & 9-\lambda \end{vmatrix} = 0$$

$$-\lambda^3 + 26\lambda^2 - 213\lambda + 560 = 0$$

$$\lambda = \{12.7, 7.0, 6.3\}$$
$$\bar{v} = \left\{ \begin{pmatrix} -0.5 \\ -0.8 \\ -0.5 \end{pmatrix}, \begin{pmatrix} -0.6 \\ 0.8 \\ 0.2 \end{pmatrix}, \begin{pmatrix} 0.2 \\ -1 \\ 0.2 \end{pmatrix} \right\}$$

These values and corresponding vectors satisfy the eigenvalue equation to with a tolerance for round off error. There are countless applications of this type of analysis including vibration analysis, finding principal axis in dynamics, principal stresses in solid mechanics, and even the principal component analysis used earlier in this work. From these applications, and others, we know that minor change in the contents of A can result in complex roots and consequently results with complex roots. For many applications this is an important and useful result, but in this case operating in the complex plane is undesirable. The above equation is the most common expression for the eigenvalue problem, but it technically defines only the right eigenvector. There exists a corresponding left eigenvector $\bar{u}$ such that $\bar{u}A = \lambda I \bar{u}$. These Eigen decompositions are generally relevant when a matrix can be thought of as a transformation or mapping of an n-dimensional space onto itself [148].

The notion that a matrix has both left and right eigenvectors can be extended to create a matrix decomposition that is generalizable to non-square matrices which are mappings from one space onto another [148]. This technique, the Singular Value Decomposition (SVD) was first presented by Beltrami and Jordan around the turn of the twentieth century [149]. Jordan began by looking for minimums and maximums of a pseudo-similarity transform of the matrix A.

$$x^{\boldsymbol{T}}Ay$$
$$||x||^2 = ||y||^2 = 1$$

A necessary condition for an extreme value is that the determinant be zero.

$$0 = dx^T Ay + x^T Ady$$

A is non-zero so the following must be true.

$$dx^T x = dy^T y = 0$$

Combing those two equations and introducing a scalar σ, Jordan obtained two equations.

$$Ay = \sigma x$$

$$x^T A = \sigma y$$

The sigma term is the maximum of the original system, so Jordon observed that it was determined by the roots of the following determinant.

$$\begin{vmatrix} -\sigma I & A \\ A^T & -\sigma I \end{vmatrix}$$

The roots of this determinant can be organized into a diagonal matrix Σ and two matrices *U* and *V* calculated from it such that *A* is decomposed as in the following equation.

$$A = U\Sigma V^T$$

Calculating *U* and *V* is not a trivial process and they are non-unique, but they can be constructed as orthogonal matrices whose columns are normalized singular vectors. Unlike the eigenvalue decomposition, it can be shown that for real *A*, all elements of the SVD are real. The decomposition can be generalized to operate on complex matrices as well by replacing the transpose of *V* with its Hermitian transform. Given the challenges in calculating *U* and *V* that satisfy those criteria for non-trivial problems, many years elapsed between the description of the SVD and initial practical applications. In 1965 Golub and Kahan introduced a stable algorithm for efficiently calculating the SVD that is the basis for most modern implementations of the SVD [148-150]. Using an implementation of Golub and Kahan's algorithm we can calculate the SVD of the example matrix used to demonstrate Eigen decomposition. That factorization is shown below.

$$A = \begin{bmatrix} 9 & 1 & 2 \\ 3 & 8 & 5 \\ 2 & 1 & 9 \end{bmatrix} = U\Sigma V^T =$$

$$\begin{bmatrix} -0.50545 & 0.8562 & 0.1072 \\ -0.6462 & -0.2933 & -0.7045 \\ -0.5718 & -0.4254 & 0.7015 \end{bmatrix} \begin{bmatrix} 13.6159 & 0 & 0 \\ 0 & 7.2252 & 0 \\ 0 & 0 & 5.6923 \end{bmatrix} \dots$$

$$\dots \begin{bmatrix} -0.5605 & 0.8270 & 0.0447 \\ -0.4588 & -0.2651 & -0.8481 \\ -0.6895 & -0.4958 & 0.5280 \end{bmatrix}^T$$

The real utility of the SVD for the applications envisioned in this dissertation is the ease with which rank reduced approximations can be calculated using it. Eckart and Young argued, and Johnson later proved, that computing the SVD of A and then retaining only the first k singular values finds the best rank-k approximation of a matrix A.

$$A_k = U_k \Sigma_k {V_k}^T$$

A and A_k are $m \times n$ matrices, U_k is a $m \times k$ matrix formed from the first k columns of U, Σ_k is a $k \times k$ diagonal matrix with the first k singular values on the diagonal, and V_k is a $n \times k$ matrix of the fist k columns of V. They went on to show that the norm of the difference between the original and rank k approximation is exactly equal to the root of the sum of the squares of the omitted singular values [151, 152].

$$\|A - A_k\| = \sqrt{\sigma_{k+1}^2 + \cdots + \sigma_r^2}$$

With only a small change to this form, we can compute the percent change due to the approximation.

$$\frac{||A - A_k||}{||A||} = \frac{\sqrt{\sigma_{k+1}^2 + \cdots + \sigma_r^2}}{||A||}$$

If we look at our sample matrix A again, we can calculate its norm as 13.6159. The magnitude of the error introduced by a rank 2 approximation would be about 40%, and a rank one approximation would be about 70%. For this small example, any approximation introduces a large error. For larger problems, significant rank reduction is often possible with minimal error. The rank 94 approximation of a rank 940 gray scale image shown in Figure 4.11 has only about 5% error compared to the original. Clearly this method of rank reduced approximation has advantages over the principal component method discussed in chapter 2 because we can now pick an approximation based on the magnitude of the error it introduces. Though it should be noted that SVD and PCA are ideas that are closely linked. A step in PCA is typically computing the SVD after the matrix of observations has been centered. Thus SVD finds the best linear subspace of a matrix, and PCA finds the best affine linear subspace. Even thought the two are so closely linked, the rest of this document will focus on rank reduction via SVD for two reasons. First, the components of the SVD are typically useful on their own, rather than recombined as they are in PCA. Query matching, in particular, can be accomplished using only a part of the decomposition. Second, there are robust techniques for incorporating new data into the SVD without computing the entire decomposition again. These folding in operations are more complicated under PCA due to the additional centering operation. Theoretically, there’s no justification for preferring SVD, but in general, it will make practical implementation of the techniques discussed easier and marginally more computationally efficient.

Finally, after weighting and calculating a rank reduced approximation, we come to the question of actually measuring the distance between objects in the vector space. There are a variety of possible distance measures, but the word distance, for many people, immediately evokes the Euclidean distance, the length of the path that connects two vectors as in Figure 4.13. In lower dimensional spaces, like Cartesian coordinates, it is relatively easy to understand the meaning of a Euclidean distance. However in higher dimensional spaces, it becomes harder to visualize with each additional dimensions. Another issue with standard Euclidean measures is the effect of the magnitude of each vector on the distance. After applying the weighting schemes and rank reduction introduced earlier, it is unclear what the real significance of each elements magnitude. Instead, a measure that looks primarily at the angular difference between vectors is preferable. We declare two observations are the same if they are co-linear and completely different if they are perpendicular, without regard to their relative magnitudes. The simplest measure that meets the criteria is the cosine distance. The cosine distance is the cosine of the angle between two vectors on the plane they define. It is calculated by taking the dot product of the two vectors and dividing by the product of their 2-norms.

$$\cos\theta = \frac{\bar{a} \cdot \bar{b}}{\|\bar{a}\|_2 \|\bar{b}\|_2}$$

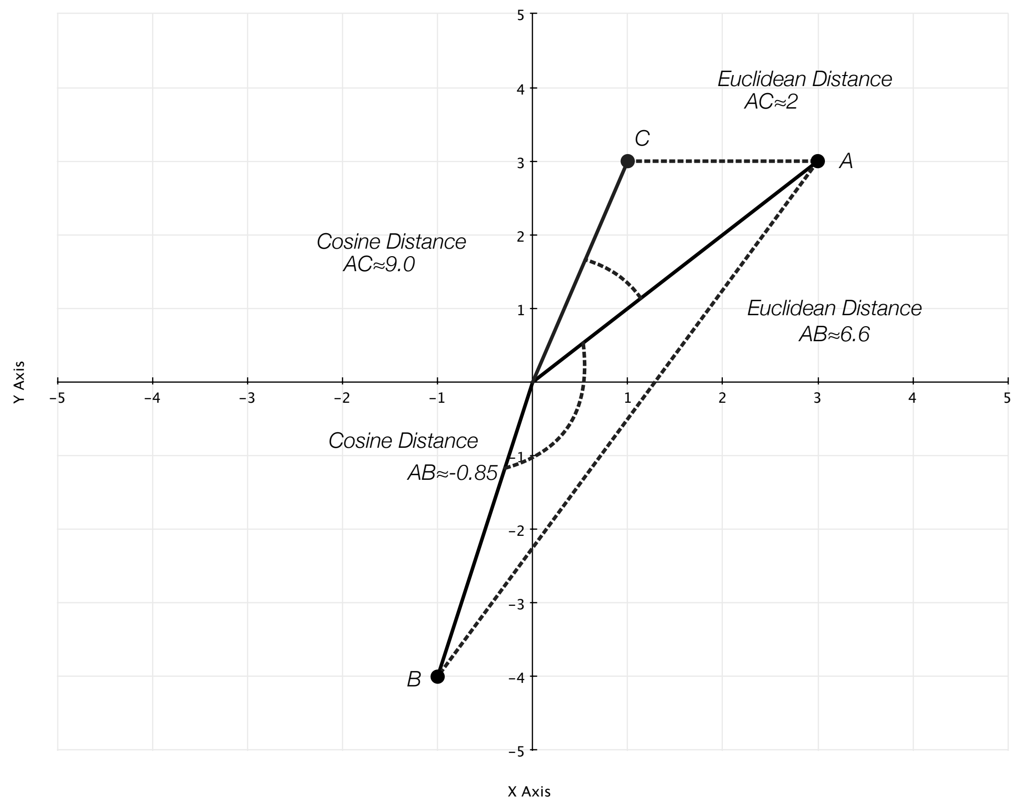


Figure 4.13 Example of Euclidean and Cosine Distance

All our observations will be positive, so cosine distance will be measured on the interval between 0 and 1, where 1 indicates that two vectors are collinear and 0 indicates they're totally orthogonal. In principal any threshold for similarity can be chosen, but in keeping with the precedent of chapter 3, measures above 0.5 will indicate similarity between two objects.

4.3.6 An Example: Text Book Title Similarity

In section 4.3.3, a term by document matrix was introduced for the titles form a set of textbooks on my bookshelf. Applying Log-entropy weighting transforms that matrix into the one shown in Table 4.2.

Table 4.2 Weighted Text Book Title Vectors

Title / Term	AED	AD	FA	MED	PD
Advanced	0.5571	0	0	0	0
Aero	0	0	0.7725	0	0
Analytical	0	0.7725	0	0	0
Design	0	0	0	0.5693	0.7847
Dynamics	0.5571	0.7725	0.7725	0	0
Engineering	0.5571	0	0	0.5693	0
Fundamentals	0	0	0.7725	0	0
Mechanical	0	0	0	0.5693	0
Product	0	0	0	0	0.7847

Following the methodology outlined above, the SVD of this term by document matrix can be calculated. The singular values of that matrix are, to two significant digits, 1.55, 1.26, 0.97, 0.87, and 0.63. The full matrix is rank 5; using Eckart and Young's formula we can find the error associated with approximating by rank reduction. This is a small matrix, large errors should result form any rank reduction. In fact we find that reducing from rank 6 to rank 5 results in an error of about 40%. This is apparently a big error, but upon actually calculating the rank 4 approximation of the term by document matrix, it's found that most of the difference occurs past the fourth significant digit. Thus, for this example the original matrix and its rank 4 approximation are essential the same. Using the cosine distance measure discussed above, the pairwise distance between book titles can be calculated. The result is summarized in the following table.

Table 4.3 Pairwise Cosine Distances of Text Book Titles

	AED	AD	FA	MED	PD
AED	1.00	0.41	0.33	0.33	0.00
AD	0.41	1.00	0.41	0.00	0.00
FA	0.33	0.41	1.00	0.00	0.00
MED	0.33	0.00	0.00	1.00	0.41
PD	0.00	0.00	0.00	0.41	1.00

Examining the results, none of the books meet a 0.5 cutoff for similarity. It seems that the first three titles *Advanced Engineering Dynamics*, *Analytical Dynamics*, and *Fundamentals of Aerodynamics* form a set of somewhat similar titles, and *Mechanical Engineering Design* and *Product Design* form another. Obviously, a small set of titles is unlikely to correctly group the results. Titles simply do not contain enough information. When we read the titles we are able to guess, probably, correctly which are similar, but that is because we have a large data set in memory to help us understand that latent meaning the words carry. To show how the process is improved by more data, I extracted the description of each book from amazon.com [153]. I transformed the text into a vector space model using the text mining plugin for R called tm [154, 155]. Using the same approach that was applied to the titles, I calculated log-entropy weights for each term and constructed a low rank approximation to the original data. A table of pairwise comparisons of cosine distance is presented in **Error! Reference source not found.**

Table 4.4 Pairwise Cosine Distances of Text Book Descriptions

	AED	AD	FA	MED	PD
AED	1.00	0.17	0.17	0.09	0.01
AD	0.17	1.00	0.96	0.55	0.05

FA	0.17	0.96	1.00	0.56	0.06
MED	0.09	0.55	0.56	1.00	0.02
PD	0.01	0.05	0.06	0.02	1.00

From the results, it is clear that a comparison based on the titles alone was not far off, even based on the descriptions the books are relatively dissimilar. *Analytical Dynamics* and *Fundamentals of Aerodynamics* seem closely related, and *Mechanical Engineering Design* is somewhat related. These three books are applied mechanics texts, so they should be deemed similar to one another, and *Product Design* focuses on the design process, so it should be an outlier. However, seems like a failure of the approach that *Advanced Engineering Dynamics* isn't found similar to the other mechanics texts. This is an artifact of the way the books' publishers have written their descriptions. The three mechanics texts that are found to be similar have descriptions that emphasize their textbook features, examples, problem sets, and solution manuals. *Advanced Engineering Dynamics* has a description that emphasizes the topics covered like kinematics and ridged body motion. Without a larger sample, the approach can't reconcile these differences and sees the omitted dynamics text as separate from the others.

This example obviously indicates that the suggested approach is not beneficial, for small data sets or for objects that can are described with a few possible parameters. Hopefully, it has shown how the basic approach advocated in this work is implemented. The following sections will validate it by tackling a series of more representative problems.

4.4 Exploring Inter-Product Functional Similarity Through the Techniques of LSI

In Chapter 3, an existing technique for finding functionally analogous products to a reference functional model was discussed in detail. Clearly finding analogies is one problem in engineering design where a vector space similarity measure is potentially useful. A measure that is applicable to a wide variety of models and encodings would be an especially useful contribution. Since there is already an approach in the literature specific to function based analogy, there is an opportunity to benchmark the proposed techniques and evaluate their appropriateness and effectiveness. The following section will construct and manipulate a vector space model of a set of existing products and their functionality following the approach outlined earlier in the chapter. Then, inter-product similarity will be measured and compared against the results of Chapter 3. The proposed method will be shown to meet or exceed the performance of the existing technique.

4.4.1 Constructing and Weighting a Vector Space Model of the Data

We begin exactly as in chapter 3 by querying the Design Repository using the SQL query supplied in Appendix A to construct a vector space model of the data. Each column represents one of the 167 products in the repository, and each row a function-flow tuple from a fixed taxonomy of function and flow. The value in each cell of the matrix represents the frequency with which that function appears in the model of each product. Here we have already begun to deviate from the approach in chapter 3. Rather than relying on our, or others', subjective judgments of importance, we are simply operating on what the data tell us. A possible objection to the approach here is that functional models are typically constructed as graphs, and by storing only the frequency with which particular function flow pairs occur, some of the interesting details about their interaction with one another is lost. The approach outlined here, could work on a vectorized form of the adjacency matrices of those graphs, but for the sake of more direct comparison with existing methods, the approach of chapter 3 is mimicked exactly. A final concern with product vectors is the inclusion of so called supporting functions. These are elements in the database, which exist to capture the physical interconnections between components, but are not part of the overall functionality of the product. The intent, as discussed in Chapter 3, is to find functional rather than structural analogies. To that end, these supporting functions are explicitly omitted in product function matrix. Once the vector space model of the data has been retrieved, it must be weighted appropriately.

For the remainder of this work, the log-entropy weighting scheme will be used for its ability to bring forward the salient properties of each product in the dataset. Proper weighting highlights elements in the data, which help to distinguish products from one another while minimizing the impact of data with little relevance. In the case of product-function vectors, heavy weight will go to functions that occur in only a few products. So a function like *import solid* that occurs in scores of products will get a small weight, while a function like *convert electrical energy to thermal energy* that occurs in a smaller subset of products will get a correspondingly higher weight. The weighted and unweighted product function matrices are attached in Appendix D.

Examination of them shows that weighting worked as intended, rare entries with low magnitude are given very high weights, while frequent entries in the original set become small in the weighted matrix.

4.4.2 Rank Reduction via Singular Value Decomposition

Following the procedure outlined earlier in the chapter, a rank reduced approximation of the original data is constructed using the singular value decomposition of the weighted product function matrix. The original matrix has rank 166, which corresponds to the number of products in the matrix. Using Eckart and Young's formula, the error introduced by each possible low rank approximation can be calculated. Figure 4.14 shows a plot of the percent error associated with approximations from rank 1 to rank 165.

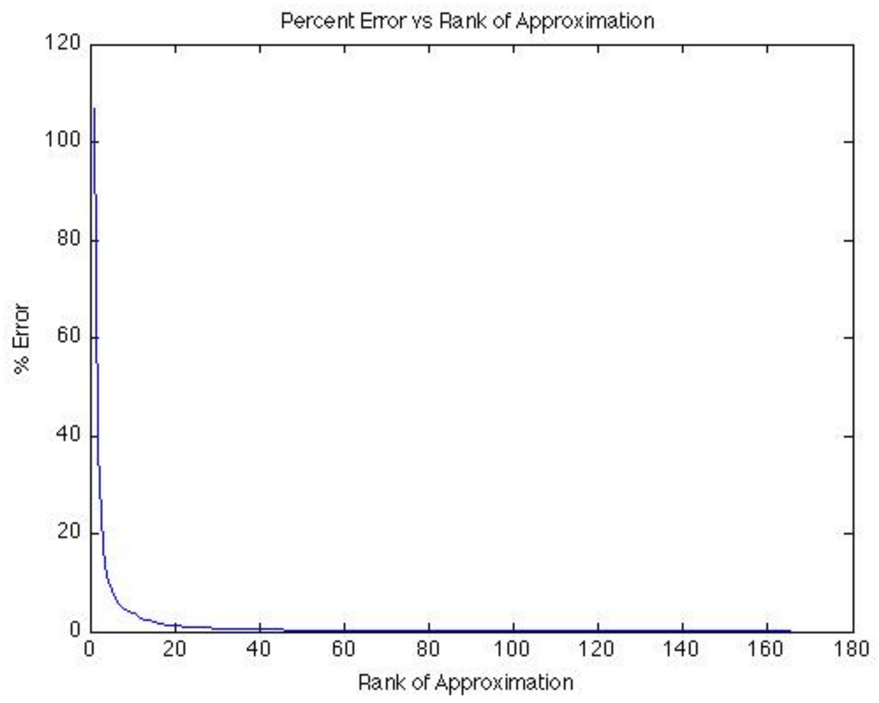


Figure 4.14 Percent Error for Rank Reduced Approximations of Weighted Product Function Matrix

From the plot, we see that the relative error between the original and rank reduced approximations is low even for very large reduction in rank. An error of 1% is acceptable, so a rank 46 approximation is used. For an error of

5% reduction to rank 12 is possible, but there's not a compelling reason to tolerate so high an error when a suitably low rank approximation can be had with much less deviation from the original data. Recall that the purpose of rank reduction is to reduce the impact of error and uncertainty in the data set. It is necessary to make enough of an approximation to mask some of the noise in the data, but not so great an approximation that useful signals in the data are also hidden. Errors on the order of a few percent have worked well for the data sets I've experimented with, though some references in the LSI literature accept much higher relative error. A reasonable amount of error will be dictated by a necessarily subjective assessment of the size and level of noise in the data set. The more data, or the more noise, present the greater the acceptable error.

4.4.3 Comparison With Existing Similarity Measure

The inter-product similarity is determined by calculating the cosine distance between each product and every other product in the dataset. As in Chapter 3, several questions will be asked of the data. First, general trends in inter-product functional similarity can be examined. In Figure 3.2, duplicated as Figure 4.15 Trends in Inter-Product Similarity Using Existing Measure, the old similarity measure found 3 types of products, a small set with few similar products, a large set with few similar products and many somewhat similar products, and a set with weak similarity to all products. Figure 4.16 is an identical plot generated using the new measure.

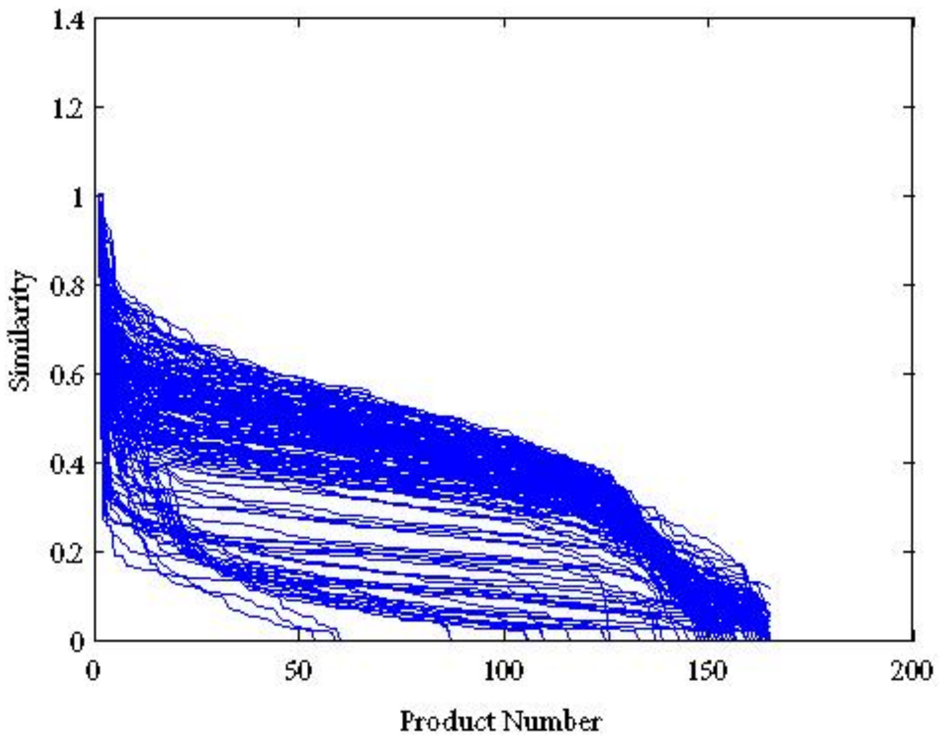


Figure 4.15 Trends in Inter-Product Similarity Using Existing Measure

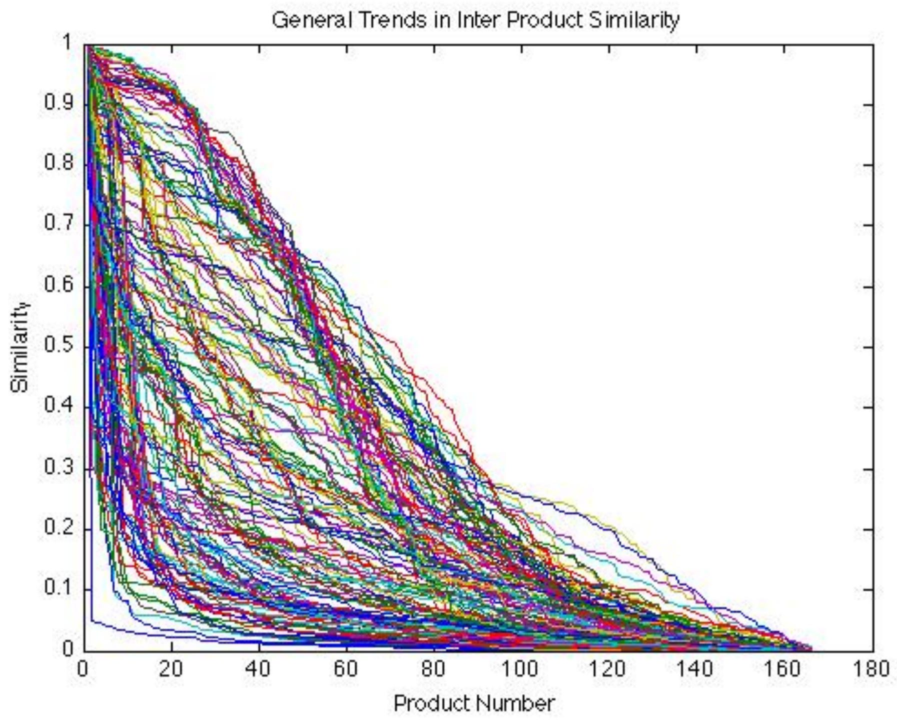


Figure 4.16 Trends in Inter-product Similarity Using Proposed Measure

The first obvious difference is the loss of the plateaus of similarity seen in the existing measure. In general similarity between each product and others in the set declines rapidly and steadily for all products, indicating that most products have a few very near neighbors functionally speaking. Second, we see that there are many more points in 0.8-1.0 range and in the 0-0.2 range. This indicates that very similar products get much higher scores and dissimilar products much lower scores using the new measure. This is desirable behavior; in the old measure many products landed around the border between similarity and dissimilarity. Under the new measure they tend to fall clearly on one side of the line.

In Chapter 3, it was shown that the existing similarity measure found a series of products that were relatively dissimilar from even their nearest neighbors. Repeating that analysis using the proposed method yields the following histogram for similarity between each product and its nearest neighbor.

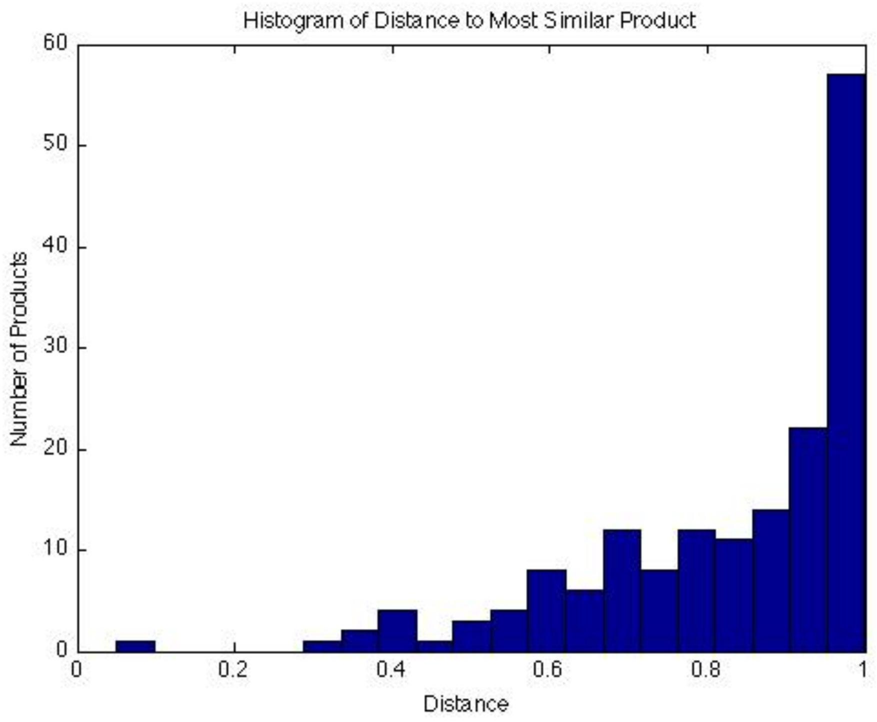


Figure 4.17 Histogram of Distance to Next Most Similar Product

Using the existing method in Chapter 3, the same histogram was produced and similarity was distributed around a 0.7. We can see from this plot that the new measure correctly assigns very high similarity to the next most functionally similar neighbor. While using the old method 0.5 was taken as a cutoff for similarity. With the new method, a much higher threshold is needed. There are still a number of products whose nearest neighbor is not very similar. We can check the results of the new method by comparing the list of products with distant neighbors to the one found in Chapter 3.

Table 4.5 Change in Most Dissimilar Artifacts Using New Measure

System	Original Position	New Position	Change
dryer	144	142	2
component basis	145	95	50
dishwasher	146	145	1
asm volume 2	147	151	-4
cotton candy machine	148	78	70
nasa anomaly	149	160	-11
hulk hands	150	21	129
alcohawk digital alcohol detector	151	26	125
coolit drink cooler	152	115	37
ub roller coaster	153	149	4
turtle	154	153	1
bolting	155	44	111
natural sensing	156	165	-9
fly	157	133	24
lawn mower	158	155	3
dna	159	50	109
jar opener	160	164	-4
john deere tractor gear	161	158	3
brake system	162	147	15
bat	163	10	153
two component regulatory system	164	140	24
lichen	165	137	28
heart	166	159	7

Table 4.5 shows the products with the most distant nearest neighbors found in chapter three, their position on the old and new lists, and the distance each moved by applying the new measure. While many of the products stay at the bottoms of both lists of products rank ordered by distance to their closest neighbor, many make significant jumps. The question now is does this indicate that the proposed measurement method is better or worse than the status quo. A reasonable conclusion is that it is an improvement specifically because of the products that make big jumps in the list. Consider for example the bat, this is a biological product that we would expect to have very low similarity to other artifacts. It was not surprising that the old measure found that it had no close functional neighbors. However the new measure finds that functionally, it's very similar to a stapler. A stapler and bat seem wholly unrelated until we look into the database. If we look at the artifact called bat, we find that it's specifically focused on the functionality of the wings, which the recorder has indicated have a function of transferring mechanical energy. Whether we think that interpretation is correct or not, if that is its function, the bat is like a stapler. We find similar rational for the big movers, the new measure, in general, seems to do a better job of sorting biological and complex systems.

If we look at a table of the products that, based on the proposed measure, are distant from their nearest neighbors, we find some interesting additions to the list.

Table 4.6 Products with Low Similarity to Other Members of the Repository Based on New Measure

System	Distance to Nearest Neighbor
game controller	0.480331623
john deere tractor gear	0.431422946
heart	0.42059038
nasa anomaly	0.416878109
iphone 3g s	0.396178105
camera	0.386033707
ge microwave	0.373093699
jar opener	0.370705683
natural sensing	0.307223227
walker	0.049976454

Recall that the distance is the cosine of the angle between the vectors that represent two products, so values as values approach 0 the products becoming increasingly different. This set of products can be said to be different from even their closet neighbors. Some of these products are on the preceding list, but some are new, and had close neighbors based on the old measure. One product that appears on this list, but was deemed similar to another product by the old measure was the iPhone. Under the previous method, the iPhone was found to be similar to a digital scale, and essentially nothing else. If we examine the data in the repository we can speculate that their similarity in the old measurement method was based on the fact that both are powered by electrical energy, and both measure some facet of the environment. In reality, it is difficult to argue that a phone and a scale are functionally similar in any real way. Based on an examination of the products at the low similarity end of the spectrum it seems that the LSI based measure improves upon existing method, and does a better job of making comparisons based on the primary functionality of each system.

In Chapter 3, four test cases were identified based on a priori beliefs about which systems should be most similar to one another. We can revisit these test cases using the new method to further compare it to the existing measure. The following table summarizes the test cases used in Chapter 3.

Table 4.7 Summary of Test Cases

Test Case	Description	Products
1	Coffee Makers	Black 12 cup deluxe, black 12 cup economy, black 4 cup regular, white 4 cup economy
2	Biological Systems	Any artifact where system type is biological
3	Aerospace Systems	A set of artifacts representing aerospace systems recorded to capture failure data
4	Drills	Skil drill, firestorm drill, delta drill, b and d drill attachement

First, we can examine the four coffee makers. The prior measure found that the four coffee makers where more similar to one another than any other products. The following table shows the four most similar products to each of the four coffee makers. This matches exactly what was found using the method discussed in Chapter 3. We can conclude from this result, that the proposed measure preforms as well as the existing method for the first test case.

Table 4.8 Coffee Maker Similarity as Measured Using Proposed Method

Reference Product	Similar Products			
	1st	2nd	3rd	4th
black 12 cup deluxe coffee	white 4 cup economy coffee	white 12 cup regular	black 12 cup economy coffee	black 4 cup regular coffee
black 12 cup economy coffee	white 12 cup regular	white 4 cup economy coffee	black 4 cup regular coffee	black 12 cup deluxe coffee
black 4 cup regular coffee	white 12 cup regular	white 4 cup economy coffee	black 12 cup economy coffee	black 12 cup deluxe coffee
white 12 cup regular	white 4 cup economy coffee	black 4 cup regular coffee	black 12 cup economy coffee	black 12 cup deluxe coffee
white 4 cup economy coffee	white 12 cup regular	black 12 cup economy coffee	black 4 cup regular coffee	black 12 cup deluxe coffee

Next, we can examine the set of biological products. In chapter 3, it was found that the existing method had mixed performance for these systems. In general it found some similarity amongst biological systems, and low similarity between biological and electromechanical systems. In Chapter 3, this was seen as a plausible result, but given that electromechanical and biological systems were encoded using the same taxonomy of function and flow, the result seemed dubious. Using the LSI based similarity measure, those biological systems that were most similar to other biological systems are now seen as very similar, but interestingly some electromechanical products have entered the list at moderate to high levels of similarity. This division is based on differences in the way biological systems were recorded. Biological systems that were recorded using the flow biological energy tend to be similar to other similarly recorded biological systems, while those that were recorded using more common terms like mechanical or electrical energy, tend to group with electromechanical systems. Based on these results, for the second test case, the proposed measure is as good as or better than the existing one.

Next, we consider the third test case, products recorded to capture failure information. These systems recorded data from ASM failure reports, Consumer Product Safety Commission Reports, and NASA and NTSB aircraft failure reports. The expectation for this test case was that these products should be relatively unique with few similar products because they represent a domain of products that is quite different from the consumer scale electro mechanical products that make up the bulk of the repository. In Chapter 3, the existing similarity measure was found to contradict this assumption. High degrees of similarity were found between some aerospace systems and consumer products, and each failure product had at least one functionally similar product. Table 4.9 shows the results of applying the proposed measure to this test case.

Table 4.9 Aerospace Systems Similarity as Measured by LSI Based Method

Reference Product	Similar Products			
	1st	2nd	3rd	4th
nasa anomaly	-	-	-	-
galileo	asm volume 1	rotorcraft	-	-
cpsc	-	-	-	-
asm volume 1	rotorcraft	galileo	robotic arm	-
asm volume 2	-	-	-	-

The results more closely match our prior expectation. Some aerospace systems have no functionally similar products in the repository as was expected. Those that are similar to other products are from within the same domain. For example Galileo systems are functionally most similar to the products extracted from ASM volume one and a set of rotorcraft. A similar result was found for ASM volume one. The products identified would all fall within the scope of large, aerospace products rather than the consumer scale products found elsewhere in the repository as was expected. Based on the results of this test case and the biological systems test case, there is preliminary evidence to suggest the proposed measure outperform the exiting one for systems which are very different from other members of the data set.

Finally, the fourth test case dealt with a set of drills. In Chapter 3, it was suggested that this might be a poor test case, because while superficially similar, when the selected products are examined closely there is not a great deal of functional similarity. That test case, when repeated using the proposed measure yields somewhat better results than were found with the existing method. The results are summarized in Table 4.10.

Table 4.10 Drill Similarity as Measured Using Proposed Method

Reference Product	Similar Products			
	1st	2nd	3rd	4th
b and d drill attachment	b and d sander attachment	b and d jigsaw attachment	crest toothbrush	bat
delta drill	mac cordless drill-driver	b and d screwdriver	digger dog	irobot roomba
firestorm drill	delta sander	b and d jigsaw	versapak sander	delta jigsaw
skil drill	skil jigsaw	delta nail gun	presto salad shooter	mini bumble ball

First, there is the Black and Decker, drill attachment. Instead of a stand-alone drill, this product is a modular head that attaches to a rotary tool. Interestingly, the proposed similarity measure indicates that is most similar functionally, to the two other modular heads for that tool. The next most similar product is an electric toothbrush that is also based on a standard body that provides rotation to a modular head. Finally there's the bat, which as discussed earlier is assigned a function of transferring mechanical energy, so it's position here is appropriate. The other drills are found to be similar, primarily to other products for which operations on rotational mechanical energy are major aspects of their functionality. This differs from the results of Chapter 3, where power tools tended to be grouped with other power tools related to cutting, or material removal. This is a manifestation of an important difference between the current and proposed measures. These are all relatively complex products and so a similarity measure could focus on various aspects of each product's functionality. The existing measure does not do a satisfactory job of emphasizing the important elements of each product's functionality. The new measure specifically addresses the value of any function in differencing a product from every other product through the log-entropy approach to weighting the data. As a result, the new measure sees rotation as being more salient that separating solid, and finds a more diverse set of related products.

4.5 *Efficient Computation of LSI Based Vector Space Similarity*

In the previous section, the functional similarity of products stored in a design repository was assessed. Applying a series of test cases indicated that a measure derived from adapting the methods of Latent Semantic Indexing outperformed an existing functional analogy tool found in the literature and discussed in detail in Chapter 3. While the preceding sections illustrate that the proposed approach works well, there are some practical issues related to a real implementation of a functional analogy search tool that can further increase its performance.

First, in Section 4.4, pairwise comparisons were made among artifacts already in a design repository. In reality, an analogy search tool would expect as an input the functional model of a product in the midst of the design process. Then, the degree of functional similarity between that model and each product in a repository would be measured to find appropriate analogous products. A first step is to convert the user generated functional model into a vector space representation suitable for comparison. Before the repository data was weighted and approximated, it existed as a set of product function vectors. A first step is to convert the user supplied functional model into a vector where each row represents the presence or absence of one of the thousands of function-flow tuples recognized by the taxonomy of function and flow used in the data set.

The data set used for comparison will be the aggregated product-function vectors for all, or perhaps a subset of, the products in a design repository appropriately weighted and rank reduced. This is an expensive computation, especially the singular value decomposition of the matrix, so; practically speaking it should be done only when new products are added to the data set and then stored for use whenever a user supplies a query.

With a query functional model, and an appropriately constructed and stored dataset, all the pieces are in place, but some additional mathematical manipulation allows us to fit everything together more efficiently. Recall that the data set was rank reduced via SVD. Instead of multiplying the rank reduced pieces back together, we can make use of some of the singular values and left and right singular vectors to speed up computation. Recall that our data matrix a can be decomposed into it's singular values and corresponding left and right singular vectors.

$$A = U\Sigma V^T$$

The rank-k approximation is given by the first k singular values, and the first k left and right singular vectors.

$$A_k = U_k\Sigma_k {V_k}^T$$

A query functional model q that is not explicitly in A is introduced. The cosine distance between any column in A_k and q is can be determined by the cosine distance formula. The vector e_j is the jth canonical vector; the jth column of the identity matrix.

$$\cos\theta_j = \frac{\left(A_k\, e_j\right)^T \cdot q}{\left\|A_k\, e_j\right\|_2 \|q\|_2}$$

Berry, in a 1999 summary of LSI techniques presents an useful transformation of that equation[126].

$$\cos\theta_j = \frac{\left(A_k\, e_j\right)^T \cdot q}{\left\|A_k\, e_j\right\|_2 \|q\|_2} = \frac{\left(U_k\Sigma_k {V_k}^T\, e_j\right)^T \cdot q}{\left\|U_k\Sigma_k {V_k}^T\, e_j\right\|_2 \|q\|_2} = \frac{{e_j}^T V_k\Sigma_k \cdot \left({U_k}^T q\right)}{\left\|\Sigma_k {V_k}^T\, e_j\right\|_2 \|q\|_2}$$

$$= \frac{\left(\Sigma_k {V_k}^T\, e_j\right)^T \cdot \left({U_k}^T q\right)}{\left\|\Sigma_k {V_k}^T\, e_j\right\|_2 \|q\|_2}$$

The left hand terms are the same for any q, so these can be pre-computed and stored. This could result in significant reduction in run time for large data sets.

4.6 Conclusion

Techniques from information retrieval can be successfully adapted to measure the distance between vector space representations of engineering design data. The results of work in this chapter provide affirmative answers to the first two research questions introduced in chapter 1. Techniques from information retrieval can be utilized to meet the need for similarity measures in engineering design, and those techniques compare favorably to existing methods. The development of and calculations necessary to support this similarity measure were discussed in detail. This approach was compared against the quantitative measure of functional analogy discussed in Chapter 3, and found to meet or exceed the performance of the of the existing measure. The new measure handles odd cases like biological and aerospace systems in a more logical and consistent manner, while mimicking the results of the existing measure for easy systems like coffee makers. Having demonstrated that the proposed measure outperforms prior work in assessing inter-product functional similarity; the pieces necessary to build a viable function based analogy search tool are introduced. The vector space similarity assessment approach introduced and validated in this chapter can now be modified to tackle the problem of guiding automated concept generation.

5 A Variety Maximizing Concept Generator

5.1 Introduction

In the first chapter, key limitations of current implementations of automated concept generators were discussed. Specifically we saw that the computational design synthesis loop posited by Cagan et al [21] of generating, evaluating, and finally guiding solutions was not closed. A way to guide the direction of new concept generation was needed. If the purpose of the concept generator is to guide the designer in their exploration of the solution space, the tool be directed to find a set of solutions with significant variety. Each newly generated concept should be as different as possible from the set already produced. In chapter two it was demonstrated that though concepts could be grouped after generation an appropriate distance measure for guiding generation was lacking. Chapters 3 and 4 laid the groundwork for such a measure by demonstrating that vector space similarity measures from information retrieval were applicable to problems in engineering design. Finally, with that work complete and two of the three original research questions satisfactorily answered, this chapter develops an automated concept generation algorithm which produces only a few distinct concept types and stops when additional concepts are not sufficiently different from those already generated. Once an improved concept generator algorithm is developed, it is validated by applying it to the test problems at the end of Chapter 2.

5.1.1 Roadmap

This chapter begins by revisiting the MEMIC algorithm for automated concept generation and explains its basic operation. Based on the results of chapter 4, a modification to MEMIC is presented which will successfully guide it to identify and return a few novel solutions from the set of all possible solutions. This eliminates the need for the complicated sampling and clustering approach of Chapter 2. The modified algorithm is then applied to the same three sample problems used for clustering in Chapter 2. The results of applying the new algorithm to these problems demonstrate that this approach can be successfully used to close the CDS loop for MEMIC like automated concept generators.

5.1.2 Contributions of This Chapter

1. Applies the vector space similarity measure of Chapter 4 to guide automated concept generation
2. Closes the CDS loop for a MEMIC like automated concept generation algorithm
3. Identifies a product component matrix from the design repository as a way to extract latent concept component information

5.2 Adapting the MEMIC Algorithm

The concept generation algorithm developed in this chapter begins with Bryant's MEMIC algorithm [20]. There are a number of reasons to prefer it as a starting point. The most obvious were that the source is readily available and it relies on the same data source, the Design Repository used in the rest of this dissertation. Beyond merely practical reasons, it's a useful starting point because its algorithm is similar to the procedure that a human designer would employ while manually generating concepts from a morphological matrix [6, 53]. This yields results that are easily interpreted, and the rationale behind the generator producing a particular concept is not hard to deduce after the fact.

Grammar based approaches, on the other hand, yield a potentially complex recipe of steps that must be understood to follow the generators rationale. Another benefit of a morphological rather than grammar based approach is the ability to discover solutions based on a set of existing products rather than a set of constructed rules. I believe there are benefits to both approaches. The best solution to automated concept generation is probably an amalgam of both, but for now MEMIC and the morphological approach to automated concept generation present the most logical starting point for a proof of concept for similarity based guidance of the concept generation process.

5.2.1 Overview of the Existing Algorithm

A full description of the MEMIC algorithm can be found by consulting Bryant [20]. The following is a high level overview of the algorithm. The process begins with the user generating a functional model, which we assume follows the taxonomy and modeling procedure of the Functional Basis [16, 17]. The adjacency matrix of its graph, which I'll call FM, can represent the functional model. The following example is the adjacency matrix for a hypothetical model with three functions.

$$FM = \begin{array}{cccc} & f1 & f2 & f3 \\ f1 & 0 & 1 & 0 \\ f2 & 0 & 0 & 1 \\ f3 & 0 & 0 & 0 \end{array}$$

Given this set of functions, it is necessary to find a set of possible components that could perform each function. Each concept will be an ensemble of these components. This information can be found by querying the design repository for a function-component matrix (FCM), this is a vector space model of a set of existing products where each column represents one of the 179 recognized component types, and each row represents a possible function. The entries correspond to the frequency with which a given component has solved particular functions. The SQL query and additional code necessary to generate an FCM is included in Appendix A and Appendix C. The following is an example FCM for the three functions in the example functional model and three hypothetical components.

$$FCM = \begin{array}{cccc} & c1 & c2 & c3 \\ f1 & 1 & 1 & 0 \\ f2 & 1 & 0 & 0 \\ f3 & 0 & 0 & 1 \end{array}$$

Possible components to insert into the adjacency matrix are calculated by multiplying rows of the FCM to produce a matrix of possible solutions. Finding a solution for the f1f2 element in the FM is possible with the following expression. The only possible solution is the first component c1.

$$f1^T f2 = [1 \quad 1 \quad 0]^T [1 \quad 0 \quad 0] = \begin{matrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix}$$

This can be repeated for each element in the adjacency matrix to develop possible concepts. MEMIC further refines the output by admitting only component chains that are feasible based on data in the design repository. If two components have not been connected in an existing product, MEMIC rejects any concept that connects them. A wrinkle for long chains is that slight variations in component order may make a concept feasible again. For example, if the chain of three components c1-c2-c3 has not been seen in an existing product, but c1-c3-c2 has, the feasibility of the concept is unclear. Bryant and others [20, 61], have proposed allowing some reordering to admit additional concepts.

5.2.2 Checking Compatibility

This error checking and component reordering presents the first problem in adapting the MEMIC algorithm. I would like to reject concepts which are not sufficiently different from those already generated while MEMIC will simultaneously reject concepts it deems infeasible. It's important in this initial validation that I know the algorithm has terminated because it's run out of novel solutions, not because it deems a particular type of solution feasible. I have consequently, chose to remove the compatibility checks from MEMIC for the purpose of this study. I justify this in a couple of ways. First, the ultimate purpose of this exercise is to generate concepts in early design where the goal is thorough exploration of the solution space. Rejecting a particular solution because it includes component connections not seen in existing products limits the variety of solutions that can be explored. Second, a reliable FCM should be sufficient to handle most compatibility issues. When I say function, I really mean a function flow pair. If a component solves a function then it admits the requisite flow. Two functions are connected by their flows, so the components that solve each operate on the same flow. They should, therefore, be compatible with one another, though possibly through an intermediate component. Based on this justification, I will omit compatibility checks form this concept generator implementation.

5.2.3 Guiding Through Rank-Reduced Vector Space Similarity

In chapter 4, it was shown that rank reduced vector space representations of products could be used to assess inter-product functional similarity in a way that emphasized the salient features of each product. My modification to the concept generator algorithm is to guide the generation of new concepts by maximizing the difference between each new concept and those that have already been generated. The similarity measurement of Chapter 4 is adapted to assess the difference between concepts based not on functionality, which is the same for all concepts, but component selection.

The first issue then is how to construct a rank reduced vector space representation of each concept's components. A non-reduced vector space model would simply be a 179-element vector where each entry indicates the use of a particular component type in the taxonomy. A more refined model would be based on component adjacency lists of each concept, but for now the simpler model is used. In Chapter 4, rank reduction was achieved through singular value decomposition of the matrix of product vectors. That is not feasible here, the decomposition would have to be recomputed each time a new concept was added, and the resultant approximations would be heavily biased toward the first concepts generated. The features of the concept component space must be captured before concepts are generated. Instead, product component vectors from the design repository can form the necessary space. Eventually, the concept will be a product; I hypothesize that reduction based on the product component space will capture appropriate dimensions of concept variation.

A product component matrix is created from the design repository using the queries and scripts in Appendix A and Appendix C. The singular value decomposition of this matrix is found by the algorithm of Golub and Kahan [150], and the error associated with potential reduced rank approximations is found using Eckhart and Young's equation [152]. Experimentation with approximations of a variety of data matrices from the Design Repository suggests that an acceptable error is between 1% and 5%. Accepting a 1% error indicates that a rank 70 approximation of the data is possible. At the end of chapter 4, it was shown that a new vector could be transformed into the rank reduced space by multiplication with the rank reduced left singular values. This property is used to construct rank reduced approximation of each generated concept.

Concept generation begins by generating an initial concept by randomly selecting a solution for each function. Next, a second concept is generated at random. This concept is then modified to produce a concept as different as possible from the first concept. For each function, a set of intermediate concepts is created that represents each possible permutation of the second concept by varying the solution to that function and holding all others constant. The distance between reduced rank approximations of these intermediate concepts and the first concept is measured following the method discussed in Chapter 4. The variant with the greatest distance is retained as the second concept and the process repeated for all the functions. The result becomes the next concept sent to the user. Given the relatively naïve search procedure, I cannot claim that this concept has the greatest possible distance from the first concept, but it is sufficiently far away to explore the boundaries of the solution space.

Each subsequent concept is generated following the same procedure as the second but with one modification. Instead of finding the alterative with the greatest distance to one concept, the alternative with the greatest distance to all concepts is the target of the search. There are a variety of ways this could be accomplished. Initially the average distance between the potential concept and the already generated concepts was used, but it tended to admit too many concepts that were neighbors of at least one other concept. Instead a metric based on the square root of the sum of the squares of the distances between the candidate concept and the already generated concept was adopted.

Finally, a termination condition was needed. The generation of new concepts should stop when additional concepts that are far away from all existing concepts cannot be generated. To achieve this, once a candidate concept has been modified to be as different as possible, I compare the distance between it the existing concepts. If the minimum distance between the candidate and any existing concept is less than a cutoff value, that candidate is rejected. Distances are cosine distances, so a minimum similarity should be somewhere between 0, totally orthogonal, to 1 exactly the same. Higher numbers will admit more concepts; lower cutoff values ensure separation between concepts. In practice, cutoffs between 0.5 and 0.8 work well. Concepts are initially generated at random, so finding one concept not sufficiently far from existing concepts does not necessarily imply that one does not exist. Instead, the algorithm is rerun with a new random seed. If it fails to find a suitable concept after trying a number of random seeds, I conclude that the algorithm is finished and a novel sample of the solution space has been found. Experience suggests that 5 rejected concepts are sufficient to ensure that a suitable concept is not overlooked. Figure 5.1 shows a schematic of the proposed algorithm.

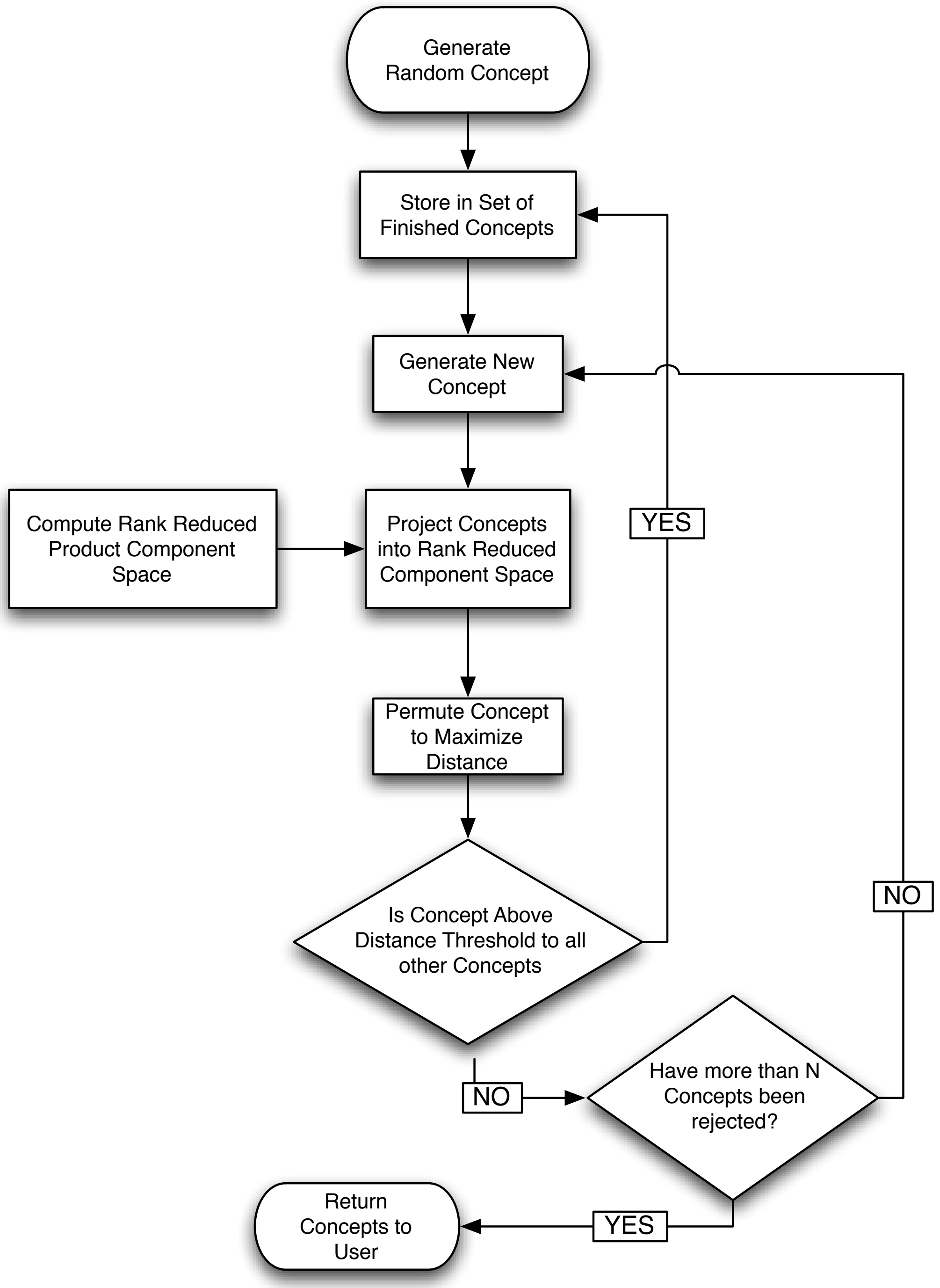

Generate Random Concept
Store in Set of Finished Concepts
Generate New Concept
Compute Rank Reduced Product Component Space
Project Concepts into Rank Reduced Component Space
YES
Permute Concept to Maximize Distance
Is Concept Above Distance Threshold to all other Concepts
NO
NO
Have more than N Concepts been rejected?
Return Concepts to User
YES

Figure 5.1 Proposed Guided Concept Generator Algorithm

The result of applying this new algorithm is a set of conceptual solutions to the design problem that are distinct from one another. This set can be shown to the designer as inspiration for further concept development and refinement. This is analogous to generating a large sample of concepts, clustering, and retrieving an exemplar of each cluster, but without the need to fully generate many concepts that will later be rejected. The following section will apply the modified concept generator algorithm to the three design problems used as test cases at the end of Chapter 2. Code that implements this algorithm is available in Appendix C.

5.3 Applying the New Algorithm to Sample Problems

Recall that at the end of Chapter 2, component based clustering of concept generator solutions was tested with three design problems. These were, a product to automatically remove the shells from peanuts as an aid to farmers in the developing world; a device to move fluid from a reservoir at one elevation to another at a higher elevation; and a consumer product to grind and dispense whole spices. For each of these problems, I review the results of Chapter 2, apply the proposed concept generator algorithm, and compare the results of both methods. In general more results should be expected due to the removal of the compatibility checks that were applied to the test cases in Chapter 2.

5.3.1 Peanut Sheller

The peanut sheller was estimated to have two unique solutions using the concept clustering technique employed in chapter two. However, both clusters had low silhouette values, based on concepts that mixed both slicing and grinding elements. The peanut sheller's functional model was input into the modified concept generation algorithm discussed in the preceding section. A similarity cutoff of 0.5 was used, any new concept that is not at a cosine distance of at least 0.5 from another generated concept is rejected. Five successive rejected components was selected as the termination condition for the algorithm. The algorithm was run 10 times; in each case the number of concepts returned was between 13 and 18 concepts. I attribute the difference to beginning with different random seeds. Fifteen concepts occurred most frequently, so the following table examines what those concepts look like.

Table 5.1 Peanut Sheller Concepts Generated Using New Algorithm

concept 1	**concept 2**	**concept 3**	**concept 4**	**concept 5**
belt	blade	material filter	divider	abrasive
shaft	shaft	nozzle	nozzle	material filter
fastener	bearing	carousel	electric conductor	nozzle
mechanical transformer	cam	shaft	shaft	housing
lever	knob	needle	sled	belt
needle	reservoir	cover	clamp	rotational coupler
cap		cam	needle	shaft
wheel		container	cushion	hinge
fan			cam	needle
support			flywheel	cam
gear			support	
sprocket			handle	
concept 6	**concept 7**	**concept 8**	**concept 9**	**concept 10**
blade	nozzle	blade	brush	nozzle
carousel	electric plug	electric conductor	nozzle	electric plug
electric plug	belt	shaft	conveyer	shaft
shaft	shaft	link	shaft	lever
fastener	lever	stop	mechanical transformer	seal

needle	cover	cam	stop	cushion
cap	seal	knob	cam	friction enhancer
cam	cam		fan	cam
knob			handle	knob
			sprocket	support
concept 11	**concept 12**	**concept 13**	**concept 14**	**concept 15**
blade	blade	nozzle	blade	blade
brush	carousel	electric plug	carousel	material filter
shaft	shaft	shaft	shaft	brush
hinge	fastener	clamp	tube	conveyer
bearing	cam	mechanical transformer	link	rotational coupler
mechanical transformer	reservoir	cam	cam	shaft
cap	flywheel	handle	knob	needle
cam	pulley	sprocket	screw	hydraulic piston
insert				cam
screw				knob
pulley				screw

The concepts derive power from human, mechanical, electrical, or hydraulic energy, and peanut shell is separated by cutting, pressure, or abrasion. While these concepts need refinement, and some of them likely could benefit from the compatibility filter typically applied by MEMIC, this approach gives a better idea of the breadth of solutions available based on repository data. A set of fifteen concepts is small enough that it can reasonably be used as a source of inspiration to the designer during concept generation. A summary of some function component replacements for this set is given in Table 5.2.

Table 5.2 Summary of Component Selection for Some Peanut Sheller Functions

Functions	**concept 1**	**concept 2**	**concept 3**	**concept 4**	**concept 5**
import solid material	cap	reservoir	cover	cushion	hinge
separate solid material	needle	blade	material filter	divider	abrasive
transfer solid material	lever	bearing	carousel	sled	nozzle
change rotational energy	gear	cam	shaft	flywheel	belt
convert rotational energy to mechanical energy	mech. Trans.	cam	cam	cam	cam
	concept 6	**concept 7**	**concept 8**	**concept 9**	**concept 10**
import solid material	cap	cover	knob	conveyer	seal
separate solid material	blade	lever	blade	brush	friction enhancer
transfer solid material	carousel	nozzle	shaft	nozzle	nozzle
change rotational energy	shaft	shaft	link	sprocket	shaft
convert rotational energy to mechanical energy	cam	cam	cam	cam	cam
	concept 11	**concept 12**	**concept 13**	**concept 14**	**concept 15**
import solid material	cap	reservoir	nozzle	tube	conveyer

separate solid material	blade	blade	clamp	blade	blade
transfer solid material	pulley	carousel	shaft	shaft	hydraulic piston
change rotational energy	Mech. Trans.	flywheel	sprocket	cam	rotational coupler
convert rotational energy to mechanical energy	cam	cam	cam	cam	cam

5.3.2 Water Lifter

In Chapter 2, concept clustering found three possible solutions to the water lifter problem. It appeared that the primary difference between concepts was the energy source employed. To compare this result to the proposed algorithm, the water lifters functional model was input as discussed in the previous section. A similarity cutoff of 0.5 and termination condition of 5 successive rejections was used again. Ten runs of the problem on the algorithm were used to counter the effects of random seed concepts. Five to eight concepts were returned each time. The difference can be attributed to beginning with different random seeds. Seven concepts occurred most frequently, so the following table examines what those concepts look like.

Table 5.3 Water Lifter Concepts Generated Using New Algorithm

concept 1	**concept 2**	**concept 3**	**concept 4**
material filter	divider	abrasive	housing
housing	blade	material filter	bearing
fan	mechanical transformer	nozzle	fastener
reservoir	lever	carousel	cap
pressure vessel	light source	cushion	screw propeller
	cam	electromagnet	pressure vessel
	container	reservoir	spring
	support	pressure vessel	
		insert	
concept 5	**concept 6**	**concept 7**	
housing	belt	blade	
rotational coupler	mechanical transformer	nozzle	
shaft	cover	electric cord	
tube	cushion	electric conductor	
seal	friction enhancer	fan	
cam	hydraulic pump	pressure vessel	
pressure vessel	container	solder	
indicator light	support	sprocket	
	solder		

Clearly these concepts do not reflect immediately buildable solutions, but they do a reasonable job of capturing the solution possibilities. Power comes from a variety of sources including electricity, mechanical energy, and fluid power. Water is moved though airfoils and fluid pressure. The new algorithm has generated a set of concepts that adequately capture the possible solution types based on function component data in the repository. These solutions can be presented to the designer as a source of inspiration and are much easier to interpret and act on than three large clusters of concepts. Table

5.4 shows some components selected for a few functions from the model for these seven concepts.

Table 5.4 Component Selections for some Functions in Water Lifter Functional Model

Functions	concept 1	concept 2	concept 3	concept 4	concept 5	concept 6	concept 7
convert mechanical energy to rotational energy	fan	cam	carousel	spring	cam	mechanical transformer	fan
transfer liquid material	fan	blade	nozzle	screw propeller	tube	hydraulic pump	pressure vessel
store liquid material	reservoir	container	reservoir	pressure vessel	pressure vessel	container	pressure vessel
export signal	housing	light source	electromagnet	housing	indicator light	cover	electric conductor

5.3.3 Spice Grinder

Finally, the spice grinder yielded poorly differentiated results when clustered with the method employed in Chapter 2. Using the same settings as the previous two problems its functional model is input into the modified concept generator algorithm. Ten runs of the algorithm yielded between 18 and 20 distinct concepts. The first ten results are shown in Table 5.5, and a summary of components selected for a subset of functions is shown in Table 5.6.

Table 5.5 Spice Grinder Concepts Generated with New Algorithm

concept 1	**concept 2**	**concept 3**	**concept 4**	**concept 5**
nozzle	brush	transistor	blade	divider
electric cord	housing	mechanical transformer	electric conductor	housing
electric plug	electric switch	cap	hydraulic pump	belt
latch release	wheel	electric motor	airfoil	link
inductor	battery	insert	knob	lever
seal	support	screen	pulley	cover
electric insulator	solder	nut-bolt		hydraulic pump
screw propeller				
pneumatic piston				
fan				
pressure vessel				
spring				
insert				
key				
screw				
concept 6	**concept 7**	**concept 8**	**concept 9**	**concept 10**
material filter	rotational coupler	housing	electric plug	sled
electric plug	tube	hinge	mechanical transformer	fuse
inductor	fuse	hydraulic pump	lever	battery
fan	hydraulic pump	knob	hydraulic pump	visual indicator
visual indicator	wheel	spring	wheel	screen
handle	support	screen	heating element	electric wire
key	handle	screw	reservoir	pulley
pulley	electric plate	pulley	pulley	
	pulley			

Table 5.6 Summary of Component Selections for some Spice Grinder Functions

Function	**concept 1**	**concept 2**	**concept 3**	**concept 4**	**concept 5**
actuate electrical energy	inductor	electric switch	transistor	knob	housing
store solid material	pressure vessel	housing	cap	hydraulic pump	cover
separate solid material	screw propeller	brush	screen	blade	divider
export solid material	nozzle	wheel	screen	blade	belt
	concept 6	**concept 7**	**concept 8**	**concept 9**	**concept 10**
actuate electrical energy	inductor	fuse	knob	lever	fuse
store solid material	material filter	tube	housing	reservoir	sled
separate solid material	key	wheel	screen	wheel	screen
export solid material	pulley	wheel	screen	wheel	screen

It is not as clear with this example what general solution types these concepts represent, but that same trend was observed with clustered concepts in Chapter 2. This is a difficult problem for the current data set. The reason why is that the core functionality of the spice grinder is to turn a monolithic solid into a set of particles. In the functional model used this functionality is represented as convert solid material to solid material. It's debatable whether or not this is a proper application of the functional basis taxonomy. Among other products in the repository this functionality is typically called "change solid" or "shape solid". The defining functionality of the proposed product is poorly captured, so the results are unfocused. This is actually a good result. If the algorithm magically changed a bad input into seemingly sensible output, something would likely be amiss. Instead we see that garbage in results in garbage out. The allegedly incorrect function is replaced with "change solid" and the results are recomputed. The number of concepts found is between 10 and 15 and variation is based on different components that change solids like blades, brushes and abrasives, and different energy sources like human, mechanical, and electrical energy. The following table summarizes the first five results from one set run using the updated functional model.

Table 5.7 Spice Grinder Concepts Using Modified Functional Model

concept 1	**concept 2**	**concept 3**	**concept 4**	**concept 5**
housing	electric cord	vibrator	blade	divider
electric conductor	electric plug	fastener	vibrator	vibrator
belt	shaft	mechanical transformer	belt	brush
rotational coupler	electric switch	lever	link	electric socket
hinge	cap	cover	transistor	thermostat
bearing	hydraulic pump	seal	inductor	lens
sled	airfoil	stop	cushion	cap
latch release	knob	hydraulic pump	friction enhancer	stop
electric resistor	spring	screen	reservoir	hydraulic pump
light source	support		circuit board	flywheel
electric motor	handle		screen	screen
wheel	screen		electric plate	key
fan				pulley
battery				
visual indicator				
insert				

5.4 Conclusion

These examples show that the vector space similarity measure introduced in chapter 4 can be used to guide automated concept generation, and answers the third research question of this dissertation. The algorithm introduced in this chapter successful closes the computational design synthesis loop for a MEMIC-like morphological matrix based automated concept generator. In contrast to the methods of chapter two, this approach rapidly identifies a set of solutions that represent the breadth of possibilities based on data in the Design Repository. The rank reduced approximation of concept vectors based on the product component space derived from the repository ensures that salient features of each concept are emphasized and variation due to uninteresting component options is minimized. While the method of clustering in Chapter 2 could occasionally amplify noisy variation amongst concepts, like the choice of a screw or a bolt to couple two pieces together, this approach buries that noise and emphasizes more meaningful variation in the set of solutions.

The proposed algorithm, while successful is only a prototype. There are many opportunities to improve its performance. Now that the approach has proven successful, compatibility checks should be reinstated. It was also noted that the number of concepts returned was not the same for multiple runs on the same problem. While the variation in number of results was small, usually one to two additional concepts, an improved algorithm would run the problem multiple times to ensure that proper number of components was returned. Finally, this algorithm needs to be combined with recent related work in the field on improving the way a MEMIC style algorithm handles functional models whose graphs have high degrees of branching [156]. These two components are major steps towards truly useful and usable automated concept generators.

6 Conclusion

This dissertation began by observing a number of limitations in current automated concept generators. The most immediate and notably of these is that they do a poor job of promoting a thorough exploration of the solutions space. Results are either a huge undifferentiated mass, or the result of the users manual exploration of the space. To realize the full potential of these tools additional work was needed. Initially, the problem was tackled by generating many concepts and clustering them into groups based on either concept parameters or the components that made up each concept. While this yielded a technique for parameter estimation that has proven useful in other work, the result was ultimately unsatisfying.

This led to the three main research questions of this work. First could a useful, widely applicable similarity measure for computational design problems be found? Second was the measure competitive with an existing ad hoc measure in its own domain? Finally, could this measure be applied to assessing concept similarity, and could concept similarity guide the generation of new concepts? Chapter 4 answered the first two questions in concert with the results of Chapter 3, while the final question is addressed by the work presented in Chapter 5.

By successfully answering these questions, a new set of beneficial tools for computational engineering design is introduced. The results bridge the cap between computational design and other information retrieval efforts. They close the computational design synthesis loop for a class of automated concept generators. Finally, they introduce an approach to extracting information from a body of existing design artifacts that can be applied to a wide variety of applications, especially analogical design and case-based reasoning.

6.1 Parameter Estimation and Concept Clustering

An initial step in improving automated concept generation was simply to take a large set of generated concepts and sort them for easier access by the user. Sorting results based on predicted concept parameters was tried initially, but was found to require a great deal of overhead work on the part of the designer as no problem independent approach could be identified. This approach yielded an estimation technique that has proven valuable in other work, and can be used by many efforts related to estimating the parameters of proposed products based on data in a design repository. Patterns observed in concepts sorted through estimated parameters suggested that component based sorting could provide the same result with less effort from the designer. Practical application of a component based sorting scheme showed the need for variable reduction and automated clustering. Unfortunately, the approach required a large sample of concepts to be generated and sorted only to discard most of the generated results. Furthermore, the algorithm can group concepts based on variations that are inappropriate or uninteresting given the problem at hand-leading to questionable results. An ideal approach to concept generation would produce just the most interesting members of the set of all possible results. Ultimately success depends on the ability of the computer to recognize salient differences in products and concepts these clustering approaches advance our ability to do that, but more work is needed.

6.2 Inter-Product Similarity a Functional Design by Analogy Perspective

Finding analogies is a useful aid to ideation in design. Currently no universal quantitative measure of analogical distance exists, but an approach was found in the literature specific to the area of function-based design by analogy. As a similarity metric was sought for the concept generation problem, it was hypothesized that this could be a useful benchmark against which future work could be tested. The measure was applied to explore inter-product similarity amongst products in the design repository. Four test cases were identified and explored. These included biological systems, coffee makers, drills, and aerospace products. For the biological system and coffee maker test cases, the measure performed as expected. For the aerospace system and drill test cases, the results contradicted initial expectations, but further examination suggests that the expectation may have been incorrect. Consequently, these presented an interesting test case. Their result defied expectation, but on later examination seemed plausible. If a new measure could handle these cases in a way that plausibly conformed to expectations that would be an interesting result. Test cases and a final rank ordered measures of product-to-product similarity formed a benchmark against which future similarity measures could be measured.

6.3 A Vector Space Similarity Measure for Engineering Design Based on Latent Semantic Indexing

It was shown that a number of useful engineering design data sources either are or can be represented as vector space models. Techniques from information retrieval were successfully adapted to measure the distance between vector space representations of engineering design data. The results provided affirmative answers to the first two research questions. A similarity measure could be adapted from information retrieval, and that measure compared very favorably to existing methods. This new approach was compared against the quantitative measure of functional analogy and met or exceeded its performance. The new measure handled odd cases like biological and aerospace systems in a more logical and consistent manner, but mimicked the results of the existing measure for common systems like coffee makers. The resulting similarity measure has a variety of applications, but this dissertation adapts it to serve as a guide in automated concept generation.

6.4 Closing the CDS Loop for MEMIC like Automated Concept Generators

A new concept generator algorithm that generates concepts representative of each major category of possible solutions was developed. The vector space similarity measures developed in this dissertation were used to guide automated concept generation. This answered the third research question of this dissertation. The computational design synthesis loop can be closed for the selected concept generator type. Unlike the sampling and clustering technique developed early in this dissertation, this algorithm rapidly identified a set of solutions that represented the breadth of possibilities based on data in the Design Repository. The rank reduced approximation of concept vectors based on the product component space derived from the repository ensured that salient features of each concept were emphasized. The focus on uninteresting component variation that plagued the earlier clustering effort is eliminated. A prototype automated concept generated based on the new algorithm was developed and tested against a set of example problems used for earlier efforts. The results show great promise in solving the problem of information overload in automated concept generation.

6.5 Contributions of this Dissertation

The following section briefly summarizes key contributions of this dissertation. I believe the following contributions represent new work with significant intellectual merit in the field of engineering design..

1. Opportunities to contribute to the state of the art in automated concept generation literature are identified
2. The need for universal approaches to design artifact similarity measurement is established
3. A method for estimating the parameters of conceptual products based on design repository data is developed
4. An approach to concept sorting based on parameter estimates is demonstrated
5. A new method for concept sorting based on component choice is developed and applied
6. Preliminary evidence that concept generator output is a set of permutations on a few solution types is found
7. An first reduced vector space representation of automated concept generator output is introduced
8. Development of four test cases for functional similarity assessment from the design repository based on current functional similarity measures
9. Construction of a benchmark for similarity assessments measures using current functional design by analogy methods and a design repository
10. Verification that many important representations of engineering design data meet the formal definition of a vector space model

11. Identification of techniques from information retrieval literature \for assessing the similarity of vector space data.
12. Application of LSI based similarity measure to functional analogy problems shows that the LSI based measure meets or exceeds the performance of the existing methods for the benchmark problems
13. Identification of the steps needed to use the LSI based similarity measure in a viable function-based analogy search tool that could replace prior work.
14. Application of an LSI based vector space similarity measure to guide automated concept generation
15. Closed the CDS loop for a MEMIC-like automated concept generation algorithm
16. Identified a product component matrix from the design repository as a way to extract latent concept component information

The chief intellectual merits of this dissertation are four fold:

1. A method for estimating parameters during conceptual design based on existing artifacts in a design repository is developed. This has already found application in other work.
2. The gulf between computational design and information retrieval is bridged. There are number of techniques in IR that may be applicable to design problems beyond those discussed in this dissertation, and this work can serve as a connection between the two fields. In collaborating with others who use repository data, we are frequently confronted with questions about how to handle noise, outliers and infrequent but important data points. The data weighting and approximation schemes of information retrieval can be an important tool to handling those problems, and this dissertation includes a first attempt at that.

3. An efficient vector space similarity measure is constructed. While this dissertation explored similarity measures primarily to advance automated concept generation, it can have other applications as well. In particular, it can be an important tool for analogical design from archives of existing products and systems, including biological systems.
4. The computational design synthesis loop is closed for at least one type of automated concept generator. This solves the problem of concept overload and allows a designer to see only a few possible solutions that represent the breadth of solutions known to the concept generator. This significantly improves the utility of automated concept generators and promotes thorough exploration of the solution space.

6.6 Impact on Designer Process and Results

Before moving on, it is important to put this body of work in perspective, and to show how it will impact the practicing design engineer. To illustrate how the results of this dissertation will improve design practice, consider a hypothetical engineer, Wallace, who has a challenging design problem to solve. Wallace's company is heavily engaged in the area of pest control, and their customers demand a new humane method for controlling infestations of small mammals, like moles and rabbits, in lawns and gardens. Wallace has some training in structured design methods, and given his understanding of his customers' needs and requirements, he develops a variety of concepts. Unfortunately, Wallace has a great deal of experience with pneumatic systems, and he fixates on pneumatic solutions to some of the key functions in his device. This leads him to select, develop, and produce a vermin vacuum system that is complex, expensive, and prone to malfunction in ways that seriously damage his customers' property. The product is a disaster and Wallace has to take a job shearing sheep to support himself.

Now, imagine that Wallace had access to the concept generation tools presented in this dissertation. In addition to his own manual concept generation efforts, Wallace inputs a functional model of his pest-capturing device into the concept generation algorithm described in Chapter 5. The computer returns a variety of concepts showcasing the solution types known to the concept generator. By analyzing this list, Wallace sees that in addition to his pneumatic concepts, there are alternatives that use bait to attract the animal and a container to hold it safely until it can be released in the wild. These results inspire Wallace to produce an entire set of alternative, non-pneumatic concepts. In concept selection he determines that these will be simpler, cheaper, and more effective than his complex vermin-vacuum. The final version of his simplified system of humane traps is a tremendous success. The more complete exploration of the design space facilitated by the tools of this dissertation helped Wallace fully explore the design space, and inspired a more optimal solution than he would have developed alone. Disaster is averted, and Wallace is handsomely rewarded.

While this is an imaginary scenario, and a tongue in cheek reference to the collected works of Aardman animation, it reflects how the tools of this dissertation can be used to improve the practice of engineering design. The tools of this dissertation enable concept generation algorithms to produce results allow the designer to see if important solution types that have been overlooked. These automatically generated solutions can inspire new concepts that would have been overlooked. Concept generation is enhanced without significant additional effort by the designer.

6.7 Future Work

The work presented in this dissertation only scratches the surface of work still to be done in the development of automated concept generators and in the other fields that this dissertation touches. There are main three areas where significant contributions can be made through continuations of this work.

First in sorting and clustering of automatically generated concepts. This work should be revisited so that it combines some of the features of the LSI based similarity tool developed later in the dissertation. Rank reduced representation of concepts instead of PCA may yield better results. K-Means clustering while widely used will increasingly be replaced by new techniques which require much less iteration and exploration to identify the actual number of clusters in the data. In particular the associative clustering of Frey and Dueck [157] is potentially very applicable to these problems and should be investigated.

Next, the similarity measures developed in this dissertation can be a useful contribution to the design by analogy literature in their own right. A more complete study should be undertaken using these methods to derive analogies to original design problems. It would be interesting to attempt to extend these methods to many dimensions of analogy beyond merely functional. In particular, it would be interesting to explore representations of product form that could be stored as vector space models and subject to this sort of analysis.

Finally, there is significant work to do to provide truly viable automated concept generators that can act as aids to ideation in the design problem. This dissertation solves one significant problem, how to identify and report a set of concepts which represent the breadth of possible solutions without overwhelming the user. The next great problem in automated concept generation is one of representation. At present, the tool returns chains or lists of components. This is not how designers communicate concepts to one another, and if the computer is to be a helpful collaborator in the design process, it has to do so on the designer's terms. Designers communicate with drawings, sketches, and other visual representations of a proposed design. The next goal for automated concept generator development should be to produce output which is more inline with this type of communication. Ideally the computer-generated concepts could be passed around the table with a concept generated by any other team member. When the research community is able to achieve this, designers will be able to move from the paradigm of computer as design aid to computer as participant in the design process. Until then, the work continues.

REFERENCES

[1] Suh, N. P., 2001, Axiomatic Design Advances and Applications, Oxford University Press, New York.
[2] Suh, N., 1998, The Principles of Design, Oxford University Press, New York.
[3] Altshuller, G., 1984, Creativity as an exact science, Gorden and Breach, Luxembourg.
[4] Hubka, V., and Ernst Eder, W., 1984, Theory of Technical Systems, Springer-Verlag, Berlin.
[5] Pahl, G., Beitz, W., Feldhusen, J., and Grote, K. H., 2007, Engineering Design a Systematic Approach, Springer-Verlag, London.
[6] Otto, K., and Wood, K. L., 2001, Product Design: Techniques in Reverse Engineering Design and New Product Development, Prentice Hall, New York.
[7] Ullman, D. G., 2010, The Mechanical Design Process, McGraw-Hill, Boston.
[8] Dym, C. L., and Little, P., 2004, Engineering Design: A Project-based Introduction, Wiley.
[9] Ulrich, K. T., and Eppinger, S. D., 2008, Product Design and Development, McGraw-Hill, New York.
[10] Szykman, S., Sriram, R. D., Bochenek, C., and Racz, J. W., 1999, "The NIST Design Repository Project," Advances in Soft Computing - Engineering Design and Manufacturing, R. Roy, T. Furuhashi, and P. K. Chawdhry, eds., Springer-Verlag, London, pp. 5-19.
[11] Bohm, M. R., Stone, R. B., and Szykman, S., 2005, "Enhancing Virtual Product Representations for Advanced Design Repository Systems," Journal of Computing and Information Science in Engineering, 5(4), pp. 360-372.
[12] Nanda, J., Simpson, T. W., Kumara, S. R. T., and Shooter, S. B., 2006, "A Methodology for Product Family Ontology Development Using Formal Concept Analysis and Web Ontology Language," Journal of Computing and Information Science in Engineering, 6(2), pp. 103-113.
[13] Szykman, S., "Architecture and Implementation of a Design Repository System," Proc. Proceedings of DETC2002.
[14] Bohm, M. R., Stone, R. B., Simpson, T. W., and Steva, E. D., "Introduction of a Data Schema: The Inner Workings of a Design Repository," Proc. ASME International Design Engineering Technical Conferences, ASME.
[15] "Design Repository," http://www.designengineeringlab.org/repository.
[16] Stone, R., and Wood, K., 2000, "Development of a Functional Basis for Design," Journal of Mechanical Design, 122(4), pp. 359-370.

[17] Hirtz, J., Stone, R., McAdams, D., Szykman, S., and Wood, K., 2002, "A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts," Research in Engineering Design, 13(2), pp. 65-82.
[18] Kitamura, Y., and Mizoguchi, R., 2003, "Ontology-based Description of Functional Design Knowledge and its Use in a Functional Way Server," Expert Systems with Application, 24(2), pp. 153-166.
[19] Kurtoglu, T., Campbell, M., Bryant, C., Stone, R., and McAdams, D., "Deriving a Component Basis for Computational Functional Synthesis," Proc. International Conference on Engineering Design, ICED'05.
[20] Bryant, C. R., 2007, "A Computational Theory for the Generation of Solutions During Early Conceptual Design," Doctor of Philosophy, University of Missouri-Rolla, Rolla.
[21] Cagan, J., Campbell, M. I., Finger, S., and Tomiyama, T., 2005, "A Framework for Computational Design Synthesis: Model and Applications," Journal of Computing and Information Science in Engineering, 5(3), pp. 171-181.
[22] Simon, H., 1993, "Anecdotes-a very early expert system," Annals of the History of Computing, IEEE, 15(3), pp. 64-68.
[23] Schon, D. A., 1992, "Designing as reflective conversation with the materials of a design situation," Research in Engineering Design, 3(3), pp. 131-147.
[24] Stiny, G., 1977, ""Ice-Ray: A Note on the Generation of Chinese Lattice Desings"," Environment and Planning B, 4, pp. 5-18.
[25] Stiny, G., and Gips, J., 1980, "Production Systems and Grammars: a Uniform Characterization," Environment and Planning B, 8, pp. 295-323.
[26] Stiny, G., and Mitchell, W. J., 1978, "The Palladian Grammar," Environment and Planning B, 7, pp. 399-408.
[27] Konig, H., and Eizenberg, J., 1981, "The Language of the Praire: Frank Lloyd Wright's Praire Houses," Environment and Planning B, 8, pp. 295-323.
[28] Stiny, G., and Mitchell, W. J., 1980, "The Grammar of Paradise: on the Generation of Mughul Gardens," Environment and Planning B, 7, pp. 209-226.
[29] Cagan, J., 2001, "Engineering Shape Grammars: Where Have We Been and Where are We Going?," Formal Engineering Design Synthesis, J. Cagan, and E. K. Antonsson, eds., Cambridge University Press, Cambridge UK.
[30] McCormack, J. P., Cagan, J., and Vogel, C. M., 2004, "Speaking the Buick language: capturing, understanding, and exploring brand identity with shape grammars," Design Studies, 25(1), pp. 1-29.
[31] McCormack, J. P., and Cagan, J., 2002, "Designing inner hood panels through a shape grammar based framework," AI EDAM, 16(04), pp. 273-290.

[32] Pugliese, M. J., and Cagan, J., 2002, "Capturing a rebel: modeling the Harley-Davidson brand through a motorcycle shape grammar," Research in Engineering Design, 13(3), pp. 139-156.
[33] Agarwal, M., and Cagan, J., 1998, "A Blend of Different Tastes: The Language of Coffee Makers," Environment and Planning B, 25, pp. 205-226.
[34] Stahovich, T. F., and Laboratory, M. I. o. T. A. I., 1995, SketchIT: A sketch interpretation tool for conceptual mechanical design, Citeseer.
[35] Stahovich, T. F., and Raghavan, A., 2000, "Computing Design Rationales by Interpreting Simulations," Journal of Mechanical Design, 122(1), pp. 77-82.
[36] Goel, A., Bhatta, S., and Stroulia, E., 1997, "Kritik: An early case-based design system," Issues and applications of case-based reasoning in design, p. 87ñ132.
[37] Stahovich, T. F., Davis, R., and Shrobe, H., 1998, "Generating multiple new designs from a sketch," Artificial Intelligence, 104(1-2), pp. 211-264.
[38] Ward, A., 1989, "A Theory of Quantitative Inference Applied to a Mechanical Design Compiler,"Doctoral, Massachusetts Institute of Technology.
[39] Ward, A. C., and Seering, W. P., 1989, "The Performance of a Mechanical Design "Compiler"," Artificial Intelligence Laboratory Massachusetts Institute of Technology, Cambridge, MA.
[40] Carlson-Skalak, S., White, M. D., and Teng, Y., 1998, "Using an evolutionary algorithm for catalog design," Research in Engineering Design, 10(2), pp. 63-83.
[41] Campbell, M. I., Cagan, J., and Kotovsky, K., 2003, "The A-Design approach to managing automated design synthesis," Research in Engineering Design, 14(1), pp. 12-24.
[42] Campbell, M., Cagan, J., and Kotovsky, K., 2003, "The A-Design approach to managing automated design synthesis," Research in Engineering Design, 14(1), pp. 12-24.
[43] Navinchandra, D., Sycara, K. P., and Narasimhan, S., 1991, "A transformational approach to case-based synthesis," Ai Edam, 5(01), pp. 31-45.
[44] Han, Y. H., and Lee, K., 2006, "A case-based framework for reuse of previous design concepts in conceptual synthesis of mechanisms," Computers in Industry, 57(4), pp. 305-318.
[45] Sycara, K., Chandra, D. N., Guttal, R., Koning, J., and Narasimhan, S., 1991, "CADET: a case-based synthesis tool for engineering design," International Journal of Expert Systems, 4(2), pp. 157-188.
[46] Welch, R. V., and Dixon, J. R., 1994, "Guiding conceptual design through behavioral reasoning," Research in Engineering Design, 6(3), pp. 169-188.

[47] Kurtoglu, T., and Campbell, M., 2009, "Automated Synthesis of Electromechanical Design Configurations from Empirical Analysis of Function to Form Mapping," Journal of Engineering Design, 20(1), pp. 83-104.
[48] Sridharan, P., and Campbell, M. l., 2005, "A study on the grammatical construction of function structures," Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 19, pp. 139-160.
[49] Campbell, M. I., Rai, R., and Kurtoglu, T., "A Stochastic Graph Grammar Algorithm for Interactive Search," ASME.
[50] Bryant, C. R., McAdams, D. A., Stone, R. B., Kurtoglu, T., and Campbell, M., "A Validation Study of an Automated Concept Generator Design Tool," Proc. ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, ASME.
[51] Bryant, C., McAdams, D., Stone, R., Kurtoglu, T., and Campbell, M., "A Computational Technique for Concept Generation," Proc. Proceedings of IDETC/CIE 2005, ASME.
[52] Bryant, C., Stone, R., McAdams, D., Kurtoglu, T., and Campbell, M., "Concept Generation from the Functional Basis of Design," Proc. International Conference on Engineering Design, ICED 05.
[53] Zwicky, F., 1969, Discovery, Invention, Research - Through the Morphological Approach, The Macmillian Company, Toronto.
[54] Ritchey, T., 1998, "General morphological analysis," A general method for non-quantified modeling, 16th EURO.
[55] Stone, R. B., Tumer, I. Y., and Van Wie, M., 2005, "The Function-Failure Design Method," Journal of Mechanical Design, 127(3), pp. 397-407.
[56] Stone, R., Tumer, I., and Stock, M., 2005, "Linking Product Functionality to Historic Failures to Improve Failure Analysis in Design," Research in Engineering Design.
[57] Tumer, I. Y., and Stone, R. B., 2003, "Analytical Methods for Mapping Function to Failure During High-Risk Component Development," Research in Engineering Design, 14(1), pp. 25-33.
[58] Vucovich, J., 2006, "The development of a functionality-centric approach to software early risk assessment," MS, University of Missouri-Rolla, Rolla, MO.
[59] Kurtoglu, T., Tumer, I.Y., 2007, "A graph based fault identification and propagation framework for functional design of complex systems," ASME Journal of Mechanical Design, 130(5), p. 8.
[60] Grantham Lough, K., Stone, R., and Tumer, I., 2009, "The risk in early design method," Journal of Engineering Design, 20(2), pp. 155-173.
[61] Parashar, T., Grantham Lough, K., and Stone, R., 2009, "The Part Count Tool (PACT) for Concept Selection," ASME 2009 International Design

Engineering Technical Conferences and Computers and Information in Engineering ConferenceSan Diego, California.
[62] Poppa, K., and Stone, R., 2009, "Sorting Results of Automated Concept Generators Based on Design for Manufacture and Assembly," ASME 2009 Internation Design Engineering Technical Converences and Computers and Information in Engineering ConferenceSan Diego, California.
[63] Boothroyd, G., Dewhurst, P., and Knight, W., 2002, Product Design for Manufacture and Assembly, Taylor and Francis.
[64] Ashby, M. F., 2005, Material Selection in Mechancial Design, Elsevier, Oxford, UK.
[65] Haapala, K. R., Poppa, K. R., Stone, R. B., and Tumer, I. Y., 2011, "Automating Environmental Impact Assessment during the Conceptual Phase of Product Design," AAAI Spring SymposieumPalo Alto, CA.
[66] Bohm, M., Haapala, K., Poppa, K., Stone, R., and Tumer, I., 2010, "Integrating Life Cycle Assessment into the Conceptual Phase of Design Using a Design Repository," Journal of Mechanical Design, 132(9).
[67] 2000, "Eco-indicator 99 Manual for Designers," S. P. a. t. E. Ministry of Housing, ed.The Hague, The Netherlands.
[68] Backer, E., 1995, Computer-Assited Reasoning in Cluster Analysis, Prentice Hall, New York.
[69] Anderberg, M., 1973, Cluster Analysis for Applications, Academic Press, New York.
[70] Orsborn, S., Boatwright, P., and Cagan, J., 2008, "Identifigying Product Shape Relationships Using Principal Component Analysis," Research in Engineering Design, 18(4), pp. 163-180.
[71] English, K., Naim, A., Lewis, K., Schmidt, S., Viswanathan, V., Linsey, J., Bishop, B., Campbell, M. I., Poppa, K., and Stone, R. B., 2010, "Impacting Designer Creativity Through IT-Enabled Concept Generation," Journal of Computing and Information Science in Engineering, 10(3).
[72] H.F., K., and Rice, J., 1974, "Little Jiffy. Mark Iv," Educational and Psychological Measurement, 34, pp. 111-117.
[73] Kaufman, L., and Rousseeuw, P. J., 1990, Finding Groups in Data: An Introduction to Cluster Analysis, Wiley, Hoboken, NJ.
[74] 2005, "The new Oxford American dictionary," Oxford University Press, New York.
[75] Otto, K., and Wood, K., 2001, Product Design: Techniques in Reverse Engineering, Systematic Design, and New Product Development, Prentice-Hall, New York.
[76] Chakrabarti, A., Sarkar, P., Leelavathamma, B., and Nataraju, B. S., 2005, "A functional representation for aiding biomimetic and artificial inspiration of

new ideas," Ai Edam-Artificial Intelligence for Engineering Design Analysis and Manufacturing, 19(2), pp. 113-132.
[77] Chiu, I., and Shu, L. H., 2007, "Biomimetic design through natural language analysis to facilitate cross-domain information retrieval," Ai Edam-Artificial Intelligence for Engineering Design Analysis and Manufacturing, 21(1), pp. 45-59.
[78] Wilson, J. O., Rosen, D., Nelson, B. A., and Yen, J., 2010, "The effects of biological examples in idea generation," Design Studies, 31(2), pp. 169-186.
[79] Sarkar, P., Phaneendra, S., and Chakrabarti, A., 2008, "Developing engineering products using inspiration from nature," Journal of Computing and Information Science in Engineering, 8(3).
[80] Casakin, H., and Goldschmidt, G., 1999, "Expertise and the use of visual analogy: Implications for design education," Design Studies, 20(2), pp. 153-175.
[81] Davies, J., Goel, A. K., and Nersessian, N. J., 2009, "A computational model of visual analogies in design," Cognitive Systems Research, 10(3), pp. 204-215.
[82] Yaner, P. W., and Goel, A. K., 2008, "Analogical recognition of shape and structure in design drawings," Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM, 22(2), pp. 117-128.
[83] Linsey, J., Wood, K., and Markman, A., "Wordtrees: A method for design-by-analogy," American Society for Engineering Education.
[84] Christensen, B. T., and Schunn, C. D., 2007, "The relationship of analogical distance to analogical function and preinventive structure: The case of engineering design," Memory & Cognition, 35(1), pp. 29-38.
[85] Huhns, M. N., and Acosta, R. D., 1998, "ARGO: A system for design by analogy," IEEE Expert, 3(3), pp. 53-68.
[86] Adelson, B., 1984, "When Novices Surpass Experts - the Difficulty of a Task May Increase with Expertise," Journal of Experimental Psychology-Learning Memory and Cognition, 10(3), pp. 483-495.
[87] Adelson, B., 1989, "Cognitive research: Uncovering how designers design; cognitive modeling: Explaining and predicting how designers design," Research in Engineering Design, 1(1), pp. 35-42.
[88] Hewett, T. T., and Adelson, B., 1998, "Psychological science and analogical reminding in the design of artifacts," Behavior Research Methods Instruments & Computers, 30(2), pp. 314-319.
[89] Linsey, J. S., Murphy, J.T., Markman, A.B., Wood, K.L., Kurtogula, T., 2006, "Representing Analogies: Increasing the Probability of Innovation," IDETC/CIE, ASME, Philadelpia, PA.

[90] Linsey, J. S., Clauss, E., Wood, K. L., Laux, J. P., and Markman, A. B., "Increasing innovation: A trilogy of experiments towards a design-by-analogy method," American Society of Mechanical Engineers, pp. 145-159.
[91] Linsey, J. S., Wood, K. L., and Markman, A. B., 2008, "Modality and representation in analogy," Ai Edam-Artificial Intelligence for Engineering Design Analysis and Manufacturing, 22(2), pp. 85-100.
[92] Linsey, J. S., Wood, K. L., and Markman, A. B., "Increasing innovation: Presentation and evaluation of the WordTree Design-by-analogy method," ASME, pp. 21-32.
[93] McAdams, D., and Wood, K., "Quantitative Measures For Design By Analogy," Proc. Proceedings of DETC2000.
[94] McAdams, D. A., and Wood, K. L., 2002, "A quantitative similarity metric for design-by-analogy," Journal of Mechanical Design, Transactions of the ASME, 124(2), pp. 173-182.
[95] Lay, D. C., 2006, Linear algebra and it's applications, Pearson/Addison-Wesley, Boston.
[96] Hefferon, J., 2011, "Linear Algebra."
[97] Steward, D., 1981, "Design Structure System: A Method for Managing the Design of Complex Systems," IEEE Transactions on Engineering Management, EM-28(3), pp. 71-74.
[98] Browning, T., 2001, "Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions," IEEE Transactions on Engineering Management, 48(3), pp. 292-306.
[99] Strawbridge, Z., McAdams, D. A., and Stone, R. B., "A Computational Approach to Conceptual Design," Proc. ASME Design Engineering Technical Conference, Design Theory and Methodology Conference, ASME, ed., ASME.
[100] Bryant, C. R., Bohm, M. R., Stone, R. B., and McAdams, D. A., "An Interactive Morphological Matrix Computational Design Tool: A Hybrid of Two Methods," Proc. ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, ASME.
[101] Poppa, K. R., Stone, R. B., and Orsborn, S., 2010, "Exploring Automated Concept Generator Output Through Principal Component Analysis," ASME 2010 International Design Engineering Technical Conferences & Computers and Information in Engineering ConferenceMontreal, Quebec, Canada.
[102] Dubin, D., 2004, "The most influential paper Gerard Salton never wrote," Library Trends, 52(4), p. 748(717).
[103] Michell, J., 1990, An Introduction to the Logic of Psychological Measurement, Lawerence Erlbaum, Hillsdale, NJ.

[104] Ginsberg, J., 2007, Advanced Engineering Dynamics, Cambridge University Press, New York.
[105] P., B. F., Johnston, E. R., and Dewolf, J. T., 2002, "Mechanics of Materials," McGraw Hill, New York.
[106] Johnson, R. A., and Wichern, D. W., 2007, Applied Multivariate Statistical Analysis, Pearson, Upper Saddle River, NJ.
[107] Alpert, J., and Hajaj, N., 2008, "We knew the web was big..." Official Google Blog.
[108] Harman, D., "Overview of the first text retrieval conference (TREC-1)."
[109] Harman, D., 1992, "The DARPA TIPSTER project," SIGIR Forum, 26(2), pp. 26-28.
[110] Singhal, A., 2001, "Modern information retrieval: A brief overview," IEEE Data Engineering Bulletin, 24(4), pp. 35-43.
[111] Salton, G., and McGill, M., 1983, Introduction to Modern Information Retrieval, McGraw Hill, New York.
[112] Salton, G., 1968, Automatic Information Organization and Retrieal, McGraw Hill, New York.
[113] Switzer, P., 1965, "Vector images in document retrieval," Statistical association methods for mechanized documentation, pp. 163–171.
[114] Sammon Jr, J. W., 1968, "Some mathematics of information storage and retrieval," DTIC Document.
[115] Tversky, A., 1977, "Features of similarity," Psychological Review, 84(4), pp. 327-352.
[116] Lalmas, M., 1999, "A model for representing and retrieving heterogeneous structured documents based on evidential reasoning," The Computer Journal, 42(7), p. 547.
[117] Theophylactou, M., and Lalmas, M., "A Dempster-Shafer model for document retrieval using noun phrases," Citeseer.
[118] Arampatzis, A., van der Weide, T. P., Koster, C., and Van Bommel, P., "An evaluation of linguistically-motivated indexing schemes," Citeseer.
[119] Jiang, F., and Littman, M. L., "Approximate dimension equalization in vector-based information retrieval," Citeseer, pp. 423-430.
[120] Salton, G., 1993, "Mathematics and information retrieval," Journal of Documentation, 35(1), pp. 1-29.
[121] Salton, G., 1989, "Automatic text processing: the transformation," Analysis and Retrieval of Information by Computer.
[122] Salton, G., Buckley, C., and Yu, C., 1983, "An evaluation of term dependence models in information retrieval," Research and Development in Information Retrieval, pp. 151-173.

[123] Salton, G., 1975, A theory of indexing, Society for Industrial Mathematics.
[124] Salton, G., Wong, A., and Yang, C. S., 1975, "A vector space model for automatic indexing," Communications of the ACM, 18(11), pp. 613-620.
[125] Salton, G., 1971, "The SMART retrieval system—experiments in automatic document processing."
[126] Berry, M. W., Drmac, Z., and Jessup, E. R., 1999, "Matrices, Vector Spaces, and Information Retrieval," SIAM Review, 41(2), pp. 335-362.
[127] Salton, G., 1971, "The SMART retrieval system—experiments in automatic document processing."
[128] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R., 1990, "Indexing by latent semantic analysis," Journal of the American society for information science, 41(6), pp. 391-407.
[129] Papadimitriou, C. H., Raghavan, P., Tamaki, H., and Vempala, S., 2000, "Latent Semantic Indexing: A Probabilistic Analysis," Journal of Computer and System Sciences, 61(2), pp. 217-235.
[130] Björck, Å., 1996, Numerical methods for least squares problems, Society for Industrial Mathematics.
[131] Chakroborty, S., and Saha, G., 2010, "Feature selection using singular value decomposition and QR factorization with column pivoting for text-independent speaker identification," Speech Communication, 52(9), pp. 693-709.
[132] Liu, K., 1997, "Application of SVD in optimization of structural modal test," Computers & Structures, 63(1), pp. 51-59.
[133] Navarro-EsbrÌ, J., Verd˙, G., Ginestar, D., and MuÒoz-Cobo, J. L., 1998, "Reactor noise analysis based on the singular value decomposition (SVD)," Annals of Nuclear Energy, 25(12), pp. 907-921.
[134] Rawat, S., Pujari, A. K., and Gulati, V. P., 2006, "On the Use of Singular Value Decomposition for a Fast Intrusion Detection System," Electronic Notes in Theoretical Computer Science, 142, pp. 215-228.
[135] Sonka, M., Hlavac, V., and Boyle, R., 1999, "Image processing, analysis, and machine vision second edition," International Thomson.
[136] Cichocki, A., and Amari, S., 2002, Blind Signal and Image Processing, Wiley Online Library.
[137] Andrews, H., and Patterson, C., 1976, "Singular value decompositions and digital image processing," Acoustics, Speech and Signal Processing, IEEE Transactions on, 24(1), pp. 26-53.
[138] Dumais, S., 1994, "Latent semantic indexing (LSI) and TREC-2," NIST SPECIAL PUBLICATION SP, pp. 105-105.

[139] Berry, M. W., Gillis, N., and Glineur, F., "Document classification using nonnegative matrix factorization and underapproximation," Institute of Electrical and Electronics Engineers Inc., pp. 2782-2785.
[140] Berry, M. W., Dumais, S. T., and Letsche, T. A., "Computational methods for intelligent information access," IEEE, pp. 390-430.
[141] Berry, M. W., Dumais, S. T., and O'Brien, G. W., 1995, "Using linear algebra for intelligent information retrieval," SIAM Review, 37(4), pp. 573-595.
[142] Letsche, T. A., and Berry, M. W., 1997, "Large-scale information retrieval with latent semantic indexing," Information sciences, 100(1-4), pp. 105-137.
[143] Pauca, V. P., Shahnaz, F., Berry, M. W., and Plemmons, R. J., "Text mining using non-negative matrix factorizations," Society for Industrial and Applied Mathematics Publications, pp. 452-456.
[144] Story, R. E., 1996, "An explanation of the effectiveness of latent semantic indexing by means of a Bayesian regression model," Information Processing & Management, 32(3), pp. 329-344.
[145] Husbands, P., Simon, H., and Ding, C., 2005, "Term norm distribution and its effects on Latent Semantic Indexing," Information Processing & Management, 41(4), pp. 777-787.
[146] Dumais, S. T., 1991, "Improving the retrieval of information from external sources," Behavior Research Methods, 23(2), pp. 229-236.
[147] MacKay, D. J. C., 2003, Information theory, inference, and learning algorithms, Cambridge Univ Pr.
[148] Moler, C. B., 2004, Numerical computing with MATLAB, Society for Industrial Mathematics.
[149] Stewart, G., 1993, "On the early history of the singular value decomposition," SIAM review, pp. 551-566.
[150] Golub, G., and Kahan, W., 1965, "Calculating the singular values and pseudo-inverse of a matrix," Journal of the Society for Industrial and Applied Mathematics: Series B, Numerical Analysis, pp. 205-224.
[151] Johnson, R. M., 1963, "On a theorem stated by Eckart and Young," Psychometrika, 28(3), pp. 259-263.
[152] Eckart, C., and Young, G., 1936, "The approximation of one matrix by another of lower rank," Psychometrika, 1(3), pp. 211-218.
[153] "Amazon.com," http://www.amazon.com.
[154] Team, R. D. C., 2011, "R: A Language and Environment for Statistical Computing," R Foundation for Statistical Computing, Vienna, Austria.
[155] Feinerer, I., Hornik, K., and Meyer, D., 2008, "Text Mining Infrastructure in R " Journal of Statistical Software, 25(5).
[156] Choudhary, A. K., and Bryant, C. A., 2010, "Automated Concept Generation Using Branched Functional Models," International Design

Engineering and Computers Conferences and Information in Engineering ConferenceMontreal Quebec.
[157] Frey, B. J., and Dueck, D., 2007, "Clustering by Passing Messages Between Data Points," Science, 315(5814), pp. 972-976.

APPENDICES

APPENDIX A SQL QUERIES

The following appendix includes the SQL queries necessary to extract the data used in this dissertation from the Design Repository. These queries reflect the schema at the time this document was written, but could be easily be made obsolete by changes to the repository's structure. Access to the database may be requested by contacting a current repository administrator at http://repository.designengineeringlab.org.

```
//--pcm_data.sql—
-- a SQL query to retrieve data needed to build a product
component Matrix
SELECT
    public.artifact.system,
    public.artifact.basis_name
FROM
    public.artifact;
```

```
//--function_hash.sql
-- a SQL query to retrieve data needed to build a hash of
function ids and function names
SELECT
    public.subfunction_type.id,
    public.subfunction_type.subfunction,
    public.subfunction_type.tier,
    public.subfunction_type.child_of_subfunction
FROM
    public.subfunction_type
ORDER BY
    public.subfunction_type.id ASC ;
```

```
//--flow_hash.sql
-- a SQL query to retrieve data needed to build a hash of flow
ids and function names
SELECT
    public.flow_type.id,
    public.flow_type.flow,
    public.flow_type.tier,
    public.flow_type.child_of_flow
FROM
    public.flow_type
ORDER BY
    public.flow_type.id ASC ;
```

```
//--component_hash.sql
-- a SQL query to retrieve data needed to build a hash of
component ids and function names
SELECT
    public.comp_basis_type.id,
    public.comp_basis_type.component,
    public.comp_basis_type.tier,
    public.comp_basis_type.child_of_component
FROM
    public.comp_basis_type
ORDER BY
    public.comp_basis_type.id ASC ;


//--fcm_data.sql
-- a SQL query to retrieve data needed to build a Function
Component Matrix

SELECT
    public.artifact.basis_name,
    public.comp_basis_type.component,
    public.function.subfunction_type,
    public.subfunction_type.subfunction,
    public.flow.input_flow,
    in_flow_type.flow,
    public.flow.output_flow,
    out_flow_type.flow,
    SUM(public.artifact.quantity)
FROM
    public.function
INNER JOIN public.artifact
ON
    (
        public.function.describes_artifact = public.artifact.id
    )
INNER JOIN public.flow
ON
    (
        public.function.id = public.flow.describes_function
    )
INNER JOIN public.comp_basis_type
ON
    (
        public.artifact.basis_name = public.comp_basis_type.id
    )
INNER JOIN public.subfunction_type
ON
    (
        public.function.subfunction_type =
public.subfunction_type.id
    )
```

```
INNER JOIN public.flow_type in_flow_type
ON
    (
        public.flow.input_flow = in_flow_type.id
    )
INNER JOIN public.flow_type out_flow_type
ON
    (
        public.flow.output_flow = out_flow_type.id
    )
WHERE
    public.function.supporting = false
GROUP BY
    public.artifact.basis_name,
    public.comp_basis_type.component,
    public.function.subfunction_type,
    public.subfunction_type.subfunction,
    public.flow.input_flow,
    in_flow_type.flow,
    public.flow.output_flow,
    out_flow_type.flow ;


//--dsm.sql
-- a SQL query to retrieve data needed to build a Design
Structure Matrix
SELECT DISTINCT
    public.artifact.basis_name,
    in_artifact.basis_name  AS inart,
    out_artifact.basis_name AS outart
FROM
    public.function
INNER JOIN public.artifact
ON
    (
        public.function.describes_artifact = public.artifact.id
    )
INNER JOIN public.flow
ON
    (
        public.function.id = public.flow.describes_function
    )
INNER JOIN public.artifact in_artifact
ON
    (
        public.flow.input_artifact = in_artifact.id
    )
INNER JOIN public.artifact out_artifact
ON
    (
        public.flow.output_artifact = out_artifact.id
);
```

APPENDIX B RUBY SCRIPT

The following appendix includes ruby scripts that were used to run queries on and return their results. This was an ugly workaround of difficulties in establishing a connection between the design repository and MATLAB®. I provide it here strictly to enable by work to be duplicated exactly, but no real implementation of these tools should rely on this approach. This script requires the rubygems add on and the sequel gem for database connection. I've sanitized connection information. Credentials can be requested through the contacts at http://repository.designengineeringlab.org .

```
#!/usr/local/bin/ruby
# query.rb - simple ruby program to read a prepared SQL query,
execute it, and write the results to a file

require "rubygems"
require "sequel"    # sequel gem handles database connection


# some details about the repository I include them here, but
they're not used until later when DBI is called
database="repository"
host="function2.mime.oregonstate.edu"
username=”*******”
password="*******"


# query is read from a text file helpfully named "query" for
now
query_file="query.txt"

#first check to see if the file exists and and is readable if
either fails the program quits
if File.exists?(query_file)
    if File.readable?(query_file)

    else
        puts("Query file is unreadable")
        Process.exit
    end

else
    puts("No query file available in working directory")
    Process.exit
end

#if above checks pass then open the query
input=File.open(query_file)
query=input.read

#connect to the repository
DB = Sequel.connect(:adapter=>'postgres', :host=>host,
:database=>database, :user=>username, :password=>password)

# test that the connection works
begin
    DB.test_connection
rescue Exception => error_msg
    puts error_msg
    Process.exit
end


# run query

begin

ds=DB.fetch(query)

rescue Exception => error_msg
    puts 'query failed'
    puts error_msg


end


#write out a text file with the results of the query
# the file is returned as "result.txt"
out_file_name="result.txt"

# first check to see if that file exists and if it does erase
it just to be safe

if File.exists?(out_file_name)
    File.delete(out_file_name)
    puts("A previous output file was found.  It will be
overwritten")
end

#now create or recreate the file
out_file=File.new(out_file_name,"w+")
```

```
# now writes the data set as a csv string
out_file.puts(ds.to_csv)


#writing's done close file
out_file.close
```

APPENDIX C MATLAB® SOURCE

This appendix contains the mat lab files used in this dissertation MATLAB®. Most of this work was done in MATLAB® for two reasons. First, it is an easy and accessible environment to prototype in, and includes built in libraries and functions that made some of the work much easier. The second reason, is that I hope by using what has become the de facto language taught to mechanical engineering students, it will be easier for those who come after me to pick up where I left off. This work was done in the version 64-bit Mac version of R2010B. The appendix is organized so that each new page is a function or script. Scripts are organized to roughly correspond to the outline of the dissertation.

Chapter 2 Scripts and Functions

```
%% cluster_memic.m
% A script to cluster automated concept generator output based
on component choice
% by Kerry Poppa

clear all; close all;
%%
%read files
files = dir('*.csv');
A=zeros(76,76,25);
for i=1:length(files)
  A(:,:,i)=csvread(files(i).name);
  i
end
clear i;
%% aggregate all concepts to indentify the part of the design
space actually
%used
B=sum(A,3);
[row,col]=find(B);
indx=[row col];
clear B;

%% Reformulate A to C such that each row of c represents a
concepts as a
%vector of the desing space found above
C=zeros(size(A,3),length(indx));
for i=1:size(A,3)
    for j=1:length(indx)
        C(i,j)=A(indx(j,1),indx(j,2),i);
    end
end
%% Reduce dimensions with PCA on C
[COEFF,SCORE,latent] = princomp(C);
PctExplained = latent' ./ sum(latent);
pctExplained= cumsum(PctExplained);

%% Scree Plot
plot(latent,'DisplayName','Scree Plot');
hold on

%% eliminate unnecessary PC's

%pick one of the next two comment the other out!
%cutoff=find(latent>=1,1,'last'); %Takes only components that
satisfy Kaiser Criterion ie Eigval>1
cutoff = input('how many principal components?'); %lets user
pick cutoff based on scree
```

```
coeff=COEFF(:,1:cutoff);
score=SCORE(:,1:cutoff);

%% cluster
k=15; %#of clusters
[IDX,clust,sumd] = kmeans(score,k);
% silhouette plot of clusters
silhouette(score,IDX);
```

Chapter 3 Scripts and Functions

```
%% mcadams_method.m
% by Kerry Poppa
% calculates similarity between systems using method of McAdams
et all

% begin with a product function matrix assume all functions are
equally
% important...ergo no weighting effect from customer needs

[pf_mat,~,~,sys_hash ]=pf_assembler;

mean_functions_per_product=sum(logical(pf_mat),1);

mean_functions=mean(mean_functions_per_product);

weights=mean_functions_per_product/mean_functions;

pf_weighted=pf_mat;
pf_weighted(logical(pf_weighted))=1;
pf_binary=pf_weighted;
for i=1:length(weights)
    pf_weighted(:,i)=pf_weighted(:,i)*weights(i);
end


% now norm the columns
pf_weighted_normed=pf_weighted;
for i=1:size(pf_weighted,2)

pf_weighted_normed(:,i)=pf_weighted(:,i)./norm(pf_weighted(:,i)
);
end

%now calculate inter product similarity
dim=size(pf_weighted_normed,2);

similarity=zeros(dim);
for i=1:dim
    ref_prod=pf_binary(:,i);
    ref_prod=ref_prod./norm(ref_prod);
    ref_prod=ref_prod';
    for j=1:dim
        similarity(i,j)=ref_prod*pf_weighted_normed(:,j);
    end
end
```

```
%% need to get a system hash to get meaningful results

query_file='query.txt';

copyfile('query_systems.txt',query_file);

!/usr/local/bin/ruby query.rb

% result should have been written to result.txt
if exist('result.txt','file')~=2
    error('no query result')
end

% now read the file should have one header line and be an int,
a string, an
% int and an int
fid=fopen('result.txt');
system_hash=textscan(fid,'%d
%s','HeaderLines',1,'Delimiter',',');
fclose(fid);

%% now set up output matrix of rank order lists of similarity

dim=size(similarity,1);
similar_products=zeros(dim);

for i=1:length(similarity)
    similarity_vector=[similarity(i,:);sys_hash'];
    similarity_vector=sortrows(similarity_vector');
    similarity_vector=flipdim(similarity_vector,1)';

    similar_products(i,:)=similarity_vector(2,:);
end

%% and repeat to replace with labels

similar_labeled=cell(dim,dim);

for i=1:dim
    for j=1:dim
        indx=find(system_hash{1,1}==similar_products(i,j));
        similar_labeled{i,j}=system_hash{1,2}{indx};
    end
end
```

```
function [pf_mat, function_hash, flow_hash, system_hash] =
pf_assembler()

%% assembles a product function matrix

%% get names of unique functions and hashes of function and
flow names and ids

[unique_funcs,function_hash,flow_hash]=function_flow_set;

pf_data=pf_data_query();

%% find the children of each function and flow...this will make
things easier later

% function
function_parent=[function_hash{1,1},function_hash{1,4}];
function_descendents={};

for i=1:length(function_parent)

children=function_parent(function_parent(:,2)==function_parent(
i,1),1);
    grandchildren=[];
    for j=1:length(children)

grandchildren=[grandchildren;function_parent(function_parent(:,
2)==children(j))];
    end
    function_descendents{i}=[children;grandchildren];
end

%flow
flow_parent=[flow_hash{1,1},flow_hash{1,4}];
flow_descendents={};

for i=1:length(flow_parent)
    children=flow_parent(flow_parent(:,2)==flow_parent(i,1),1);
    grandchildren=[];
    for j=1:length(children)

grandchildren=[grandchildren;flow_parent(flow_parent(:,2)==chil
dren(j))];
    end
    flow_descendents{i}=[children;grandchildren];
end


%% create a system hash
```

```
system_hash=unique(sort(pf_data(:,1)));

%% populate data

% this is a pretty naive way to solve this problem...will try
to improve
% later
pf_mat=zeros(length(unique_funcs),length(system_hash));
for i=1:length(unique_funcs)


function_set=[unique_funcs(i,1)];%;function_descendents{unique_
funcs(i,1)}];

in_flow_set=[unique_funcs(i,2)];%;flow_descendents{unique_funcs
(i,2)}];

out_flow_set=[unique_funcs(i,3)];%;flow_descendents{unique_func
s(i,3)}];

    rows_to_keep=pf_data((ismember(pf_data(:,2),function_set) &
...
        ismember(pf_data(:,3),in_flow_set) &
ismember(pf_data(:,4),out_flow_set)),:);
    for j=1:length(system_hash)

pf_mat(i,j)=sum(rows_to_keep(rows_to_keep(:,1)==system_hash(j),
5));
    end
end
```

```
function [file_exists]=existence_check(files_to_check)

% function accepts cell array of file_names and returns a
logical aray by
% checking to see if they exist



file_exists=zeros(1,max(size(files_to_check)));

for i=1:max(size(files_to_check))
    if exist(files_to_check{1,i},'file')==2
        file_exists(i)=1;
    end
end
```

```
function
[unique_funcs,function_hash,flow_hash]=function_flow_set()

%% function flow query  this file gets a list of distinct
functions and flows

%% first check to verify that the expected query files are
availabe

distinct_functions='query_distinct_functions.txt';
function_list='query_function_list.txt';
flow_list='query_flow_list.txt';

files_to_check={distinct_functions,function_list,flow_list};
indxs=find(~existence_check(files_to_check));

if indxs
    missing_file_string='';
    for i=1:length(indxs)
        missing_file_string=[missing_file_string,'
',files_to_check{indxs(i)}];
    end

    error(['Missing the following files','
',missing_file_string]);

end
%% query for function hash

query_file='query.txt';

copyfile(function_list,query_file);

!/usr/local/bin/ruby query.rb

% result should have been written to result.txt
if exist('result.txt','file')~=2
    error('no query result')
end

% now read the file should have one header line and be an int,
a string, an
% int and an int
fid=fopen('result.txt');
function_hash=textscan(fid,'%d %s %d
%d','HeaderLines',1,'Delimiter',',');
fclose(fid);

%% query for flow hash
```

```
query_file='query.txt';

copyfile(flow_list,query_file);

!/usr/local/bin/ruby query.rb

% result should have been written to result.txt
if exist('result.txt','file')~=2
    error('no query result')
end

% now read the file should have one header line and be an int,
a string, an
% int and an int
fid=fopen('result.txt');
flow_hash=textscan(fid,'%d %s %d
%d','HeaderLines',1,'Delimiter',',');
fclose(fid);

%% now get distinct function flow set
query_file='query.txt';

copyfile(distinct_functions,query_file);

!/usr/local/bin/ruby query.rb

% result should have been written to result.txt
if exist('result.txt','file')~=2
    error('no query result')
end

% now read the file should have one header line and be 3
columns of
% integers
fid=fopen('result.txt');
funcs=cell2mat(textscan(fid,'%d  %d
%d','HeaderLines',1,'Delimiter',',')); %cell2mat is because
textscan reads as cell array
fclose(fid);


%% now handle aggregating up functions and flows

%separate cases 1st where inflow=outflow

two_tuple=funcs(funcs(:,2)==funcs(:,3),1:2);

%2nd where flows don't match
```

```
three_tuple=funcs(funcs(:,2)~=funcs(:,3),:);

%need parent child relationships
funct_parent=[function_hash{1,1},function_hash{1,4}];
flow_parent=[flow_hash{1,1},flow_hash{1,4}];
%% aggregate the flow matching case

%assign data to temp variable
agg=two_tuple;

while find(agg)
    funct_up=agg;
    flow_up=agg;

  for i=1:length(agg)

funct_up(i,1)=funct_parent((funct_parent(:,1)==funct_up(i,1)),2
);

flow_up(i,2)=flow_parent((flow_parent(:,1)==flow_up(i,2)),2);
  end
  both_up=[funct_up(:,1),flow_up(:,2)];

  agg=[funct_up; flow_up; both_up];

  %get rid of zeros...i.e. flow with no agg up
  agg(agg(:,1)==0,:)=[];
  agg(agg(:,2)==0,:)=[];
  agg=unique(agg,'rows');

  two_tuple=[two_tuple; agg];
end

two_tuple=unique(two_tuple,'rows'); %get rid of any duplicates
that may have crept in

%% now the case where flows don't match
agg=three_tuple;

while find(agg)
    funct_up=agg;
    in_up=agg;
    out_up=agg;

    for i=1:length(agg)

funct_up(i,1)=funct_parent((funct_parent(:,1)==funct_up(i,1)),2
);
    in_up(i,2)=flow_parent((flow_parent(:,1)==in_up(i,2)),2);
    out_up(i,3)=flow_parent((flow_parent(:,1)==out_up(i,3)),2);
    end

    func_in_up=[funct_up(:,1), in_up(:,2), agg(:,3)];
    func_out_up=[funct_up(:,1), agg(:,2), out_up(:,3)];
    flows_up=[agg(:,1), in_up(:,2), out_up(:,3)];
    all_up=[funct_up(:,1), in_up(:,2), out_up(:,3)];

    agg=[funct_up; in_up; out_up; func_in_up; func_out_up;
flows_up; all_up];

    %get rid of zeros...i.e. aggs that spilled over the top of
the
    %heirarchy and duplicates
    agg(agg(:,1)==0,:)=[];
    agg(agg(:,2)==0,:)=[];
    agg(agg(:,3)==0,:)=[];
    agg=unique(agg,'rows');

    three_tuple=[three_tuple; agg];
end

three_tuple=unique(three_tuple,'rows'); % gets rid of
duplicates that may have crept in

%% now stitch back together

unique_funcs=unique([[two_tuple,two_tuple(:,2)];three_tuple],'r
ows');
```

```
function [pf_data]=pf_data_query(excluded_systems)

%validate inputs
if nargin<1
    excluded_systems=false;
else

    if ~isrow(excluded_systems) %ensure row vector
        error('expected a row vector of excluded systems')
    end


end

% return Product function list based on user parameters


% define the output file
output_file='query.txt';
%% first establish if the query is restricted to a number of
systems

if excluded_systems
    restricted=true;
else
    restricted=false;
end

%% handle the restricted vs unrestricted cases

switch restricted
    case true
    query_file_name='query_with_restriction.txt'; %specify of
restricted query file

    % check to make sure this file still exists in the
directory (exist
    % should report 2 if it does)
        if exist(query_file_name,'file')~=2
        error('query file not found');
        end

    %list the ids of systems (products) to include
    systems=excluded_systems;

    % convert that list to a string (matlab converts the array
with a
    % double space between numbers...so the regexp is to make a
comma separated list)
```

```
system_string=['[',regexprep(num2str(systems),'\s\s',','),']'];

    %now open the file...and rewrite the query to retrieve only
the listed systems
    fid=fopen(query_file_name);
    query_string=fread(fid,'*char')';
    fclose(fid);

    %add in system string
    query=regexprep(query_string,'{}',system_string);

    %write out to query file
    fid=fopen(output_file,'w+');
    fprintf(fid,query);
    fclose(fid);

    otherwise
    query_file_name='query_unrestricted.txt';
    %check to make sure this file still exists in the directory
(exist
    % should report 2 if it does)
        if exist(query_file_name,'file')~=2
            error('query file not found');
        end
    copyfile(query_file_name,output_file);

end

%% now execute the query

%check to make sure query file exist
if exist('query.rb','file')~=2
   error('query script missing');
end
%execute the query
!/usr/local/bin/ruby query.rb

%% result should have been written to result to txt
if exist('result.txt','file')~=2
    error('no query result')
end

pf_data=importdata('result.txt',',');

pf_data=pf_data.data;
```

Chapter 4 Scripts and Functions

```
function [similar_labeled, similar_products,
similarity]=Vector_Space_method(weighting)


%% calculates similarity between systems using my method
adapted from IR

% begin with a product function matrix assume all functions are
equally
% important...ergo no weighting effect from customer needs

[pf_mat,~,~,sys_hash ]=pf_assembler;


%% pick weighting scheme and caculate weighted matrix


switch weighting
    case {'none'}
        disp('Local and Global Weights were not applied')

        %Local weight
        % non-local weight

        %Global weight
        % just one for this case

        pf_weighted=(pf_mat);

    case {'log'}
        disp('Local and global weights calculated as log of
frequency and log entropy')

        %Local weight
        local_weight=log2(1+pf_mat);
        %Global weight
        p=pf_mat./repmat(sum(pf_mat,2),1,size(pf_mat,2));

        p_log=p.*log2(p);
        p_log(isnan(p_log))=0; % have to do this because log
blows up for terms with 0 frequency

        global_weight=1+(sum(p_log,2)./log2(size(pf_mat,2)));
        global_weight=repmat(global_weight,1,size(pf_mat,2));
        pf_weighted=pf_mat.*local_weight.*global_weight;

    otherwise
            error('weighting scheme not recoginized try "none"
or "log"')
end


%% now perfrom queries

similarity=zeros(size(pf_mat,2));
%will pre_calculate 2 norm of columns of data matrix

col_norms=sqrt(sum((pf_weighted.^2),1));

for i=1:size(pf_mat,2)

    query_vector=pf_mat(:,i);
    query_vector_norm=sqrt(query_vector'*query_vector);

    for j=1:size(pf_mat,2)

similarity(i,j)=pf_weighted(:,j)'*query_vector/(col_norms(j)*qu
ery_vector_norm);
    end

end

%% need to get a system hash to get meaningful results

query_file='query.txt';

copyfile('query_systems.txt',query_file);

!/usr/local/bin/ruby query.rb

% result should have been written to result.txt
if exist('result.txt','file')~=2
    error('no query result')
end

% now read the file should have one header line and be an int,
a string, an
% int and an int
fid=fopen('result.txt');
system_hash=textscan(fid,'%d
%s','HeaderLines',1,'Delimiter',',');
fclose(fid);
```

```
%% now set up output matrix of rank order lists of similarity

dim=size(similarity,1);
similar_products=zeros(dim);

for i=1:length(similarity)
    similarity_vector=[similarity(i,:);sys_hash'];
    similarity_vector=sortrows(similarity_vector');
    similarity_vector=flipdim(similarity_vector,1)';

    similar_products(i,:)=similarity_vector(2,:);
end

%% and repeat to replace with labels

similar_labeled=cell(dim,dim);

for i=1:dim
    for j=1:dim
        indx=find(system_hash{1,1}==similar_products(i,j));
        similar_labeled{i,j}=system_hash{1,2}{indx};
    end
end
```

```
%%FM_processor.m
%modified concept generator algorithm
%by Kerry Poppa

%% start out with a fixed file for now
fm_file='spices2.txt';


%% call FM_reader to process function adjaceny matrix
[names, FM]=FM_reader(fm_file);

%% need to reformat individual functions to make it easier to
sort fcm

names=lower(names); %ensure everything is lower case
names=regexprep(names,'\s{2,}',' '); % and single spaced



%% generate an FCM and a set of text function labels

[fcm_mat,labels,components]=new_FCM();

%% find rows of FCM that correspond to parts of functional
model

matches=[];
for i=1:length(names)
    row=find(strcmpi(names(i),labels));
    if isempty(row)
        row=0;
    end
    matches=[matches;row];
end

fcm_keep=[];

for i=1:length(matches)
    if matches(i)~=0
       fcm_keep(i,:)=fcm_mat(matches(i),:);
    elseif matches(i)==0;
        fcm_keep(i,1)=47;
    end

end
```

```
%% now find columns....these will be components

comp_ids={};

for i=1:size(fcm_keep,1)

    comp_ids{i}=find(fcm_keep(i,:));
end

%% excluded components
exclude=[1:39,175:179]; %set the components to exclude...could
be done programmatically later

for i=1:length(comp_ids)
    comp_ids{i}(ismember(comp_ids{i},exclude))=[];
end

%% if any function has no solution include unknown
for i=1:length(comp_ids)
    if isempty(comp_ids{i})
        comp_ids{i}=1;
    end
end

%% get an intial concept

[first_concept, first_cv]=solveit(comp_ids);

%% retrieve repository product component matrix for creating
reduced approximations
pcm=PCM();
[U,S,~]=svd(pcm);

error=fliplr(sum(S));
error=(sqrt(cumsum(error.^2)))./norm(pcm);
error=100*fliplr(error);

indx=find(error<1,1,'first'); %allow 1% error % find first
instance of this

Uk=U(:,1:indx);

%% create a container for finished concepts

finished_vectors=first_cv;
finished_vectors_k=Uk'*first_cv;
```

```
finished_concepts=first_concept;

%% create additional concepts
repeat=1;  %will keep going until 5 concepts in a row are non
unique
while repeat<25;
[next_concept,next_cv]=solveit(comp_ids);

%now choose a concept that maximizes distance to first

for i=1:length(next_concept)

%for each row of the fcm create a new concept with each
alternate component
%keep the one that is furthes away
original_comp=next_concept(i);
concept_mod=next_cv;
concept_mod(original_comp)=concept_mod(original_comp)-1;
possible_concepts=repmat(concept_mod,1,length(comp_ids{i}));

  for k=1:size(possible_concepts,2)

possible_concepts(comp_ids{i}(k),k)=possible_concepts(comp_ids{
i}(k),k)+1;

  end
  possible_concepts_k=[];
  for j=1:size(possible_concepts,2)
      possible_concepts_k=[possible_concepts_k,
Uk'*possible_concepts(:,j)];
  end

distances=squareform(pdist([possible_concepts_k,finished_vector
s_k]','cosine'));
  cols=size(finished_vectors_k,2);
  rows=size(possible_concepts_k,2);
  distances=distances(1:rows,size(distances,2)-cols:end);
  dist_metric=sqrt(sum(distances.^2,2));
  [~,keep_index]=max(dist_metric);
  concept_to_keep=possible_concepts(:,keep_index);
next_cv=concept_to_keep;
end


check=1-squareform(pdist([finished_vectors_k,
Uk'*next_cv]','cosine'));
check=abs(check)>0.5; %cutoff for similarity
check=sum(check);

if max(check)>1;
```

```
next_cv=[];
repeat=repeat+1;
else
finished_vectors=[finished_vectors, next_cv];
finished_vectors_k=[finished_vectors_k, Uk'*next_cv];
end
end
```

```
function [func_names, connections]= FM_reader(file_name)
%% reads a functional model adjacency matrix from FunctionCAD


file_struct=importdata(file_name);

%if importdata succeds it will put the function adjacency
matrix into a
%struct of a char array of names and a matrix of conncetions

func_names=file_struct.textdata(:,1);
% there will be one emptycell
func_names=func_names(2:end);

%the actual adjacency matrix
connections=file_struct.data;

end
```

```
function [fcm_mat,row_labels,col_labels] =  new_FCM()

%% this script prepares an FCM from queries of the repository



%% locate query files

%file to look in for FCM data query
%format should be component ID, component name, function,
function name
%input flow, flow name, output flow, flow name count
fcm_query='fcm_data.sql';

%files to look in for function hash, flow hash, and component
hash
%format should be id, name, tier, child of

function_hash_query='function_hash.sql';

flow_hash_query='flow_hash.sql';

component_hash_query='component_hash.sql';

%create cell array of names
file_names={fcm_query; function_hash_query; flow_hash_query;
component_hash_query};

%check to verify that these files exist
existence_check=cellfun(@exist,file_names);

%throw an error if any of the files don't exist - would be nice
to be
%specific about which one, but for now let's just throw the
error  (we know
%an error exists returns comethign other than 2).

assert(isequal(existence_check,2*ones(size(existence_check))),.
..
    'one of the query files was missing');

%% execute queries

%create an function that is a composite of our query and reader
functions

hash_query = @(query_file) hash_reader(db_query(query_file));
```

```
%run the hash queries
function_hash=hash_query(function_hash_query);
flow_hash=hash_query(flow_hash_query);
component_hash=hash_query(component_hash_query);

%run the data query

data_file=db_query(fcm_query);

%open the file
fid=fopen(data_file);

%read the file
fcm_data=textscan(fid,'%f %s %f %s %f %s %f %s
%f','HeaderLines',1, 'Delimiter', ',');

%close the file
fclose(fid);

%turn data matrix into an array for easier manipulation cols
are: function,
%inflow, outflow, comp, count

datamat=cell2mat(fcm_data(:,[3; 5; 7; 1; 9]));

%% aggregation
% the following is probably the most naive way I can imagine
doing this,
% but it works...

%first find the unique functions in the data set

unique_functions=unique(datamat(:,1:3),'rows');

%two cases flows match or they don't

two_tuple=unique_functions(unique_functions(:,2)==unique_functi
ons(:,3),:);
three_tuple=unique_functions(unique_functions(:,2)~=unique_func
tions(:,3),:);

%% two tuple case is easiest, do it first  three options
function_up, flow_up both up

up_once=(two_tuple(:,1:2)); %matrix to hold aggretion up one
level
up_twice=zeros(size(up_once)); %matrix to hold aggregation up
two levels
```

```
%find each function and flows parents
function_parent=function_hash{4};
flow_parent=flow_hash{4};

%do the aggregation cases
for i=1:length(up_once)

   up_once(i,1)=function_parent(up_once(i,1));
   up_once(i,2)=flow_parent(up_once(i,2));

   if up_once(i,1)~=0
   up_twice(i,1)=function_parent(up_once(i,1));
   end

   if up_once(i,2)~=0
   up_twice(i,2)=flow_parent(up_once(i,2));
   end
end

%aggregate  scenarios - will define some intermediate variables
to make it
%less confusing - this is a terrible way to do this, but I
needed the
%book keeping to make sure everything is included

func=two_tuple(:,1); flow=two_tuple(:,2); func_u=up_once(:,1);
flow_u=up_once(:,2);
func_uu=up_twice(:,1); flow_uu=up_twice(:,2);

agged_two_tuple=[func, flow;
                 func, flow_u;
                 func, flow_uu;
                 func_u, flow;
                 func_u, flow_u;
                 func_u, flow_uu;
                 func_uu, flow;
                 func_uu, flow_u;
                 func_uu, flow_uu];




%get rid of duplicates
agged_two_tuple=unique(agged_two_tuple,'rows');

%get rid of zeros

agged_two_tuple(agged_two_tuple(:,1)==0 |
agged_two_tuple(:,2)==0, :)=[];

%% now do three tuple case

up_once=(three_tuple(:,1:3)); %matrix to hold aggretion up one
level
up_twice=zeros(size(up_once)); %matrix to hold aggregation up
two levels

%do the aggregation cases
for i=1:length(up_once)

   up_once(i,1)=function_parent(up_once(i,1));
   up_once(i,2)=flow_parent(up_once(i,2));
   up_once(i,3)=flow_parent(up_once(i,3));

   if up_once(i,1)~=0
   up_twice(i,1)=function_parent(up_once(i,1));
   end

   if up_once(i,2)~=0
   up_twice(i,2)=flow_parent(up_once(i,2));
   end

   if up_once(i,3)~=0
   up_twice(i,3)=flow_parent(up_once(i,3));
   end
end

%aggregate
func=three_tuple(:,1); inflow=three_tuple(:,2);
func_u=up_once(:,1); inflow_u=up_once(:,2);
func_uu=up_twice(:,1); inflow_uu=up_twice(:,2);
outflow=three_tuple(:,3); outflow_u=up_once(:,3);
outflow_uu=up_twice(:,3);

col1=[repmat(func,9,1);repmat(func_u,9,1);repmat(func_uu,9,1)];
col2=repmat([repmat(inflow,3,1);repmat(inflow_u,3,1);repmat(inf
low_uu,3,1)],3,1);
col3=repmat([outflow;outflow_u;outflow_uu],9,1);

agged_three_tuple=[col1,col2,col3];

%get rid of duplicates
agged_three_tuple=unique(agged_three_tuple,'rows');

%get rid of zeros
agged_three_tuple(agged_three_tuple(:,1)==0 |
agged_three_tuple(:,2)==0 | agged_three_tuple(:,3)==0, :)=[];
```

```
%% put everything back together

unique_functions=[aged_two_tuple,
aged_two_tuple(:,2);aged_three_tuple];

%may still be some duplicates
unique_functions=unique(unique_functions,'rows');

%get rid of some of the temp variables created along the way to
avoid
%confusion

clear col1 col2 col3 flow flow_u flow_uu func funch_u func_uu
inflow inflow_uu inflow_u
clear outflow outflow_u outflow_uu aged_three_tuple
aged_two_tuple two_tuple three_tuple
clear up_once up_twice
%% now have a list of known functions from the repository can
deal out values to an fcm

fcm_mat=zeros(length(unique_functions),length(component_hash{1}
));

for i=1:length(unique_functions)



funcs=[unique_functions(i,1);children_finder(unique_functions(i
,1),function_hash{4})];

inflows=[unique_functions(i,2);children_finder(unique_functions
(i,2),flow_hash{4})];

outflows=[unique_functions(i,3);children_finder(unique_function
s(i,3),flow_hash{4})];

rows_to_keep=datamat(ismember(datamat(:,1),funcs) &
ismember(datamat(:,2),inflows)...
    & ismember(datamat(:,3),outflows),4:5);

    for j=1:size(rows_to_keep,1)
        row=i; col=rows_to_keep(j,1);

        fcm_mat(row,col)=fcm_mat(row,col)+rows_to_keep(j,2);
    end
end

%% one more aggregation is needed b/c of component hierarchy
```

```
for i=1:size(fcm_mat,2)
    comp_children=children_finder(i,component_hash{4});

    for j=1:size(comp_children,1)
        fcm_mat(:,i)=fcm_mat(:,i)+fcm_mat(:,comp_children(j));
    end
end


%% now we need to develop text lables for the functions

function_labels=cell(size(unique_functions,1));

%there's a problem in the flow labels, for whatever reason
human energy and
%material don't follow the same convention as other flows, so
we need to
%strip their qualifiers

flow_hash{2}=regexpi(flow_hash{2}, '^(\S+)','match');  %this
keeps only the first word in each string

% now run through the list and add the appropriate labels

top_level_indxs=find(flow_hash{4}==0);

for i=1:length(top_level_indxs)

descendents=children_finder(top_level_indxs(i),flow_hash{4});
    append=top_level_indxs(i);
    for j=1:length(descendents)
        indx=descendents(j);
        flow_hash{2}{indx}=[cell2mat(flow_hash{2}{indx}), ' ',
cell2mat(flow_hash{2}{append})];
    end
end

%% we have the right set of labels just need to compose row and
column labels
labels={};
for i=1:length(unique_functions)
    func=(function_hash{2}{unique_functions(i,1)});
    if unique_functions(i,2)==unique_functions(i,3)
          flow=(flow_hash{2}{unique_functions(i,2)});
          label=[func,' ',flow];
    else
        inflow=(flow_hash{2}{unique_functions(i,2)});
        outflow=(flow_hash{2}{unique_functions(i,3)});
```

```
            label=[func,' ',inflow,' to ',outflow];
        end

        if iscell(label)
            label=cell2mat(label);
        end
        labels{i}=label;

    end

    row_labels=labels';
    col_labels=component_hash{2}';
```

```
function [hash]=hash_reader(data_file)
% parses the results of database querys to return hashes of
taxonomies in
% the repository

%% open the file
fid=fopen(data_file);

%% read the data (will be id, name, tier, child of)
hash=textscan(fid,'%f %s %f
%f','HeaderLines',1,'Delimiter',',');

%deal with Nan's in cols 3 and 4
hash{3}(isnan(hash{3}))=0;
hash{4}(isnan(hash{4}))=0;
%%close the file
fclose(fid);
```

```
function [result_file]=db_query(query_file)
%this function calls a ruby script to query the database

%% first check to ensure the needed ruby script is missing
assert(exist('query.rb')==2,'query script is missing');

%% copy the specified query file to the file expected by the
ruby script
output_file='query.txt';
copyfile(query_file,output_file);

%% run the query
!/usr/local/bin/ruby query.rb

%% tell matlab where to read the result
result_file='result.txt';
```

```
%% generate solutions
function [concept,concept_vector]=solveit(comp_ids)
solution=zeros(length(comp_ids),1); % pre-allocate an array to
hold the solution

% check to see if any function has a single solution

for i=1:length(comp_ids)
    if length(comp_ids{i})==1
        solution(i)=comp_ids{i}(1);
    end
end

% now go through and populate the solution

while prod(solution)==0 %while there's still a zero in the set

    % pick a function at random function to start with
    already_solved=1;

    while already_solved==1

        funct=random('unid',length(solution));
        if solution(funct)==0
           already_solved=0;
        end

    end
    %pick a solution from that function

solution(funct)=comp_ids{i}(random('unid',length(comp_ids{i})))
;



end


%% reformat into concept component matrix

soln=zeros(179,1);
for i=1:length(solution)

    soln(solution(i))=soln(solution(i))+1;
end
concept=solution;
concept_vector=soln;
end
```

APPENDIX D SAMPLE PROBLEMS

The following appendix includes adjacency matrices of the functional models used to generate concepts for the Peanut Sheller, Water Lifter, and Spice Grinder in Chapter 2 and Chapter 5. These models have some imperfections as discussed in Chapter 5, but they should enable the work in this dissertation to be duplicated.

	import solid material	import human energy	export solid material	export solid material	separate solid material	convert solid material to solid material	transfer solid material	guide solid material	store solid material	distribute solid material	store solid material	guide solid material	store solid material	transfer solid material	convert human energy to rotational energy	distribute rotational energy	convert rotational energy to translational energy	change rotational energy	change rotational energy	change rotational energy	convert rotational energy to mechanical energy
import solid material	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
import human energy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
export solid material	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
export solid material	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
separate solid material	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
convert solid material to solid material	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
transfer solid material	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
guide solid material	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
store solid material	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
distribute solid material	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
store solid material	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
guide solid material	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
store solid material	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
transfer solid material	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
convert human energy to rotational energy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
distribute rotational energy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0
convert rotational energy to translational energy	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
change rotational energy	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
change rotational energy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
change rotational energy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
convert rotational energy to mechanical energy	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Peanut Sheller Functional Model

	import energy	convert mechanical energy to rotational energy	change rotational energy	transfer liquid material	actuate rotational energy	import SIGNAL	export SIGNAL	distribute liquid material	transfer liquid material	position liquid material	supply liquid material	export liquid material	export acoustic energy	export thermal energy	export mechanical energy	convert energy to mechanical energy	store liquid material
import liquid material	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
import liquid material	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
import energy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
convert mechanical energy to rotational energy	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
change rotational energy	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
transfer liquid material	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
actuate rotational energy	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
import SIGNAL	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
export SIGNAL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
distribute liquid material	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
transfer liquid material	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
position liquid material	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
supply liquid material	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
export liquid material	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
export acoustic energy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
export thermal energy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
export mechanical energy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
convert energy to mechanical energy	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
store liquid material	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Water Lifter Functional Model

	actuate electrical energy	store solid material	import solid material	secure solid material	regulate electrical energy	import control signal	convert electrical energy to mechanical energy	convert electrical energy to mechanical energy	transport solid material	separate solid material	store solid material	convert solid material to solid material	mix solid material	sense solid material	indicate status signal	export solid material	guide solid material	export energy
import electrical energy	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
import electrical energy	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
actuate electrical energy	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
store solid material	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
import solid material	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
secure solid material	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
regulate electrical energy	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
import control signal	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
convert electrical energy to mechanical energy	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
convert electrical energy to mechanical energy	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
transport solid material	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
separate solid material	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
store solid material	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
convert solid material to solid material	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
mix solid material	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
sense solid material	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
indicate status signal	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
export solid material	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
guide solid material	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
export energy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Spice Grinder Functional Model