Spring 2009

Information archival and reuse: drawing conclusions from the past

Matt R. Bohm
Missouri University of Science and Technology, mbohm@mst.edu

Department: Mechanical and Aerospace Engineering

INFORMATION ARCHIVAL AND REUSE:

DRAWING CONCLUSIONS FROM THE PAST

by

MATT ROBERT BOHM

A DISSERTATION

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE & TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

MECHANICAL ENGINEERING

2009

Approved by

Robert Stone, Advisor
Xiaopingu Du
Robert Landers
Ming Leu
Ralph Wilkerson

PUBLICATION DISSERTATION OPTION

Sections of this dissertation have been submitted for publication in relevant engineering conference proceedings and journals and have been prepared in the style utilized by these publications.

Pages 13-45 have been prepared in the style preferred by the Journal of Computer Aided Design.

Pages 46-76 have been prepared in the style preferred by the ASME International Design Engineering Technical Conferences.

Pages 77-96 have been prepared in the style preferred by the Journal of Computer and Information Science in Engineering.

Pages 97-121 have been prepared in the style preferred by the ASME International Design Engineering Technical Conferences.

Pages 122-151 have been prepared in the styled preferred by the ASME International Mechanical Engineering Congress and Expo.

The remaining pages have been prepared in accordance with the Missouri University of Science and Technology dissertation specifications.

ABSTRACT

Over the last few decades design researchers have put forward theories and proposed methodologies that increase the chance that a design team will reliably arrive at the optimal solution to a given design problem. Studies, however, bear out that theories and methodologies alone will not guarantee an optimal or even good design solution. Instead, a breadth of knowledge across multiple engineering domains and the time and tools to thoroughly evaluate the design space are as important as any prescriptive design method. This work presents a set of underlying engineering technologies to define, archive and reuse product design knowledge to provide a breadth of domain knowledge for designers and to leverage artificial intelligence approaches to thoroughly, if not exhaustively, search the design space. Specifically, a database schema and entry application for a prototype design repository of product design knowledge is formulated and implemented. A real-time, knowledge base-driven, function-based conceptual design algorithm known as the morphological search is formulated to extract information from the design repository and support a thorough exploration of the design space for solutions. Currently, the Design Engineering Lab's prototype Design Repository contains design knowledge for over 125 products and has over 300 user accounts representing 17 different countries.

With the foundational repository elements in place, artificial intelligence methods are employed to generate a natural language to formal component naming terms thesaurus as part of a novel form-initiated concept generation approach. The approach, known as Form Follows Form, automatically generates a functional model based upon an initial component solution seed to a design problem. With a functional model in hand, established automated concept generation algorithms are employed to return more complete and varied solutions following a thorough search of the design space.

ACKNOWLEDGMENTS

I would like to thank first and foremost my parents, Roger and Bobbie Bohm, they have always been there for me and given me encouragement my whole life. I would secondly like to thank my advisor, Dr. Robert Stone, for allowing me to work on the repository project and for his personal encouragement as a friend and teacher.

I would also like to thank Dr. Xiapingu Du, Dr. Robert Landers, Dr. Ming Leu, and Dr. Ralph Wilkerson for serving on my committee.

A special thanks also goes out to all of my fellow graduate and undergraduate researchers in the Design Engineering Lab. Their support and collaboration has been invaluable.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

1.1 PHILOSOPHICAL STATEMENT

The broad vision of this research is to capture existing or expert design knowledge and build tools that allow the novice, student, or less experienced designer access to expert information in an intuitive and easy to use manner. Design knowledge capture has taken form in connected database schema and design repository and the tools provide foundations in automated concept generation and reasoning. This dissertation seeks to answer the hypothesis that computational intelligence (i.e., product design knowledge archival and reuse and AI algorithms) can be applied in the early phases of design to increase the quantity, quality, and breadth of concept variants produced during the design of a product.

1.2 PROBLEM SCOPE

This research aims to link expert knowledge from not only the fields of mechanical engineering and engineering design, but also engineering and science in general. To support today's complex products, processes, and needs it is import to bridge the gaps that sometimes exists between traditional domains of study. The field of engineering is almost synonymous with design. Arguably, the essence of engineering is applying the theories and principles of the sciences to serve the needs of humankind. Stated more succinctly, engineering is focused on designing solutions to observed needs. Within engineering many fields exist to support those activities and engineering design is the one field that is concerned with the principles, theories and methodologies of design that transcend all disciplinary engineering fields. Often this field of study finds its academic home in mechanical engineering even though it is an interdisciplinary activity cutting across all of engineering.

The basic process of engineering design can generally be described as four phases that 1) clarify a problem; 2) generate conceptual solutions; 3) embody the chosen

concept; and finally 4) detail out the design for production. The input to this four phase process of design is an unmet societal need and the ultimate output is a product meeting the societal need. While depicted in Figure 1.1 as a sequential process, the process is invariable a iterative activity within each phase and between phases.

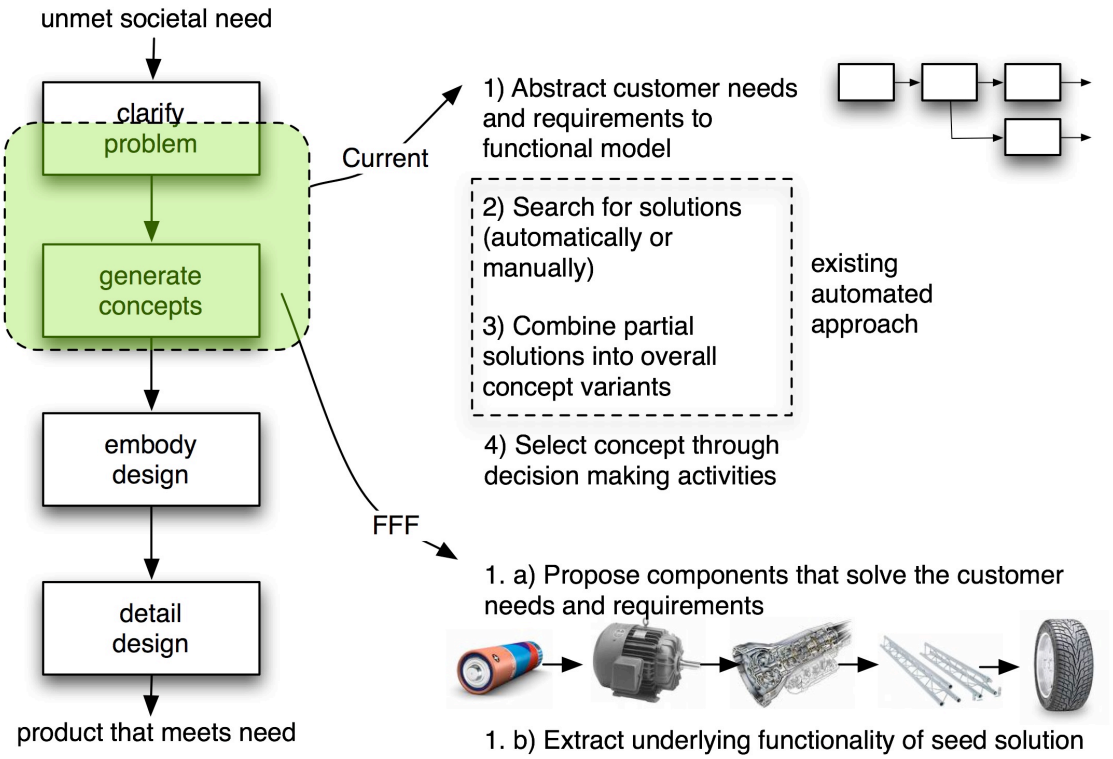


Figure 1.1. Overview of the design process (left side). Early advances in automating the early phases of design (upper right quadrant) and one major focus of the current research on moving to a form-initiated, computational thinking-based concept generation approach (lower right quadrant)

Specific to the hypothesis under consideration here, the scope of this research encompasses some latter stage activities of the clarify problem phase and most of the generate concept phase. In recent years, researchers have made progress at automating portions of the generate concept phase by introducing algorithms that transform a

functional description of the societal (or product) need into form - expressed as concept variants. The fundamental steps of this process are shown in the upper right quadrant of Figure 1.1. This advance relies on design knowledge archival methods (some of which is covered in this dissertation) and supports a more thorough, if not exhaustive, search of the solution space. One of the keys to successful operation of these algorithms is the development of the functional description of the product under consideration. In this work, that required input will be referred to as a functional model or function structure. Experience shows that abstracting the product need to a functional model is a difficult process for engineering designers. A significant portion of this research is involved with moving toward a form-initiated design approach, outlined in the lower right quadrant of Figure 1.1, that will overcome this last major stumbling block.

1.3 THE SUPERVAC - A MOTIVATIONAL CASE

Looking ahead to the potential impact of a new form-initiated approach, consider how a typical design experience in the future, informed by an approach that boosts a designer's creativity and accessible by any engineer, might play out. Audrey and Zeke work for ACME AirGas, a company that makes industrial grade vacuums and blowers. They have an identified requirement to redesign their "SuperVac" model so that it is less noisy. They are assigned to the manufacturing division and typically do not "do design." However, the design division is swamped with a new product launch and they are tasked with the redesign. The customer need of "less noise" is easily associated with an automobile muffler component by Audrey and Zeke as shown in Figure 1(a). It is, of course, not feasible to install an automotive muffler on even the industrial SuperVac model. However, by using their new form-initiated computational design tool, the vacuum's original components plus the imagined muffler are input. The underlying functionality of the components is determined (shown in Figure 1(b)) and component solutions for the functionality are returned from a query to existing function-initiated concept generation algorithms (depicted in Figure 1(c)). Based on the ability to start

with a specific, yet infeasible, component, more feasible concept models are recalled from the repository and presented to stimulate further creative modifications as represented in Figure 1(d). As a result of using the new form inspired automated concept generation, both Audrey and Zeke are able to find useful and creative ways to solve the product's noise issue (Figure 1.2). In such new product research and development scenarios, several alternatives can be pursued throughout detailed design.

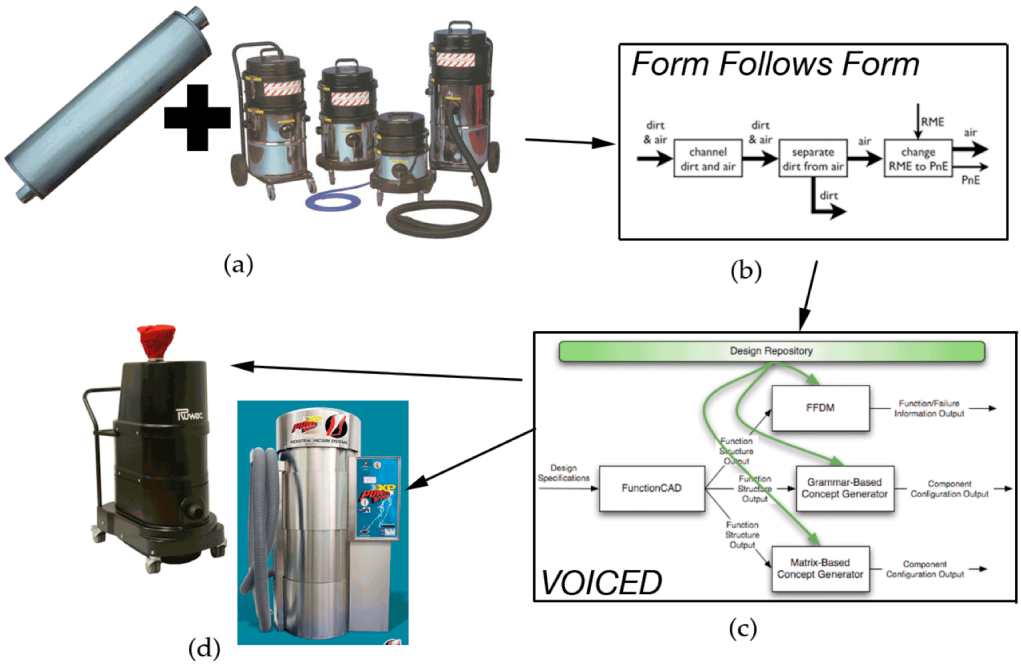


Figure 1.2. A schematic of a future form-initiated computational design tool

1.4 STATE OF THE ART

In this section a brief review of related works is presented. The topics presented include design information archival as well as manual and automated concept generation techniques.

1.4.1 NIST Repository Initiative

The most similar system to UMR's repository schema takes form in the NIST Design Repository representation model [1-5]. Through their repository initiative, NIST set out to define basic guidelines of a Design Repository and how archived design information could be useful to designers. The NIST Design Repository representation model is a basic framework to help guide what type of product information is collected and how the elements of information are related to each other. NIST has also developed a mapping from this representational framework into an XML (eXtensible Markup Language) data format. While portions of the NIST initiative overlaps with design representation standards such as STEP (Standard for the Exchange of Product model data), the breadth and scope of implementation differ greatly. Like certain STEP protocols, the NIST framework provides for geometric and process information storage but also expands them to a higher-level domain of design information storage.

The NIST initiative proposes a set of information models to be used for modeling product knowledge at varying levels of detail. There are several data entities which allow for a variety of aspects of a product description to be represented. The classes specified in the NIST Core Product Model include: Artifact, Function, Transfer Function, Flow, Form, Geometry, Material, Behavior, Specification, Configuration, Relationship, Requirement, Reference and Constraint.

1.4.2 PDM Systems

In recent years product data management (PDM) systems have emerged to help store and retrieve product and part data. PDM systems allow for part hierarchy storage as well as process data and project management elements. Svensson and Malmqvist [6] explore a PDM system and demonstrate many uses of such a system. The PDM system demonstrated collects requirement, function, concept and part structures as well as property models. Additionally a PDM system stores the entire product structure, variants, revisions and finally documentation and CAD models. Although function

structures and property models can be stored within a PDM system, they are not capable of storing the detailed function based information we desire and integrating it into useful design tools without heavy modification. A PDM system is a highly effective tool for use in the manufacturing side of emerging products and parts but is fundamentally different from a repository system.

1.4.3 The Design Repository

The objective of a Design Repository is to allow designers to store and retrieve design knowledge at various levels of abstraction, from form (components, sub-assemblies and assemblies) to architecture description to function. Currently the Missouri S&T Design Repository contains design information for over 125 consumer based electro-mechanical products. Design information captured by the repository can be divided into seven main categories including: artifact-, function-, failure-, physical-, performance-, sensory- and media-related information types. The different levels of abstraction and types of design information provide innovative ways to approach design. With a well populated repository, emerging concept generator algorithms take, as input, basic product functionality or component information and instantaneously develop, filter and rank concepts to use as baselines for further product development. While the possibilities design repositories offer are diverse and helpful to designers, the implementation of such repositories are crucial to their overall success and usefulness.

Realizing the potential impact of an operational Design Repository, researchers at Missouri S&T, The University of Texas at Austin and the National Institute of Standards and Technology (NIST) began gathering artifact information in 1999 [7-9]. Since that time, the process in which artifact data is gathered and recorded has changed significantly. Initially, artifact design information was recorded in spreadsheets and mainly took the form of Bills of Materials (BOM), Function Component Matrices (FCM), and Design Structure Matrices (DSM). While this type of information was useful, it was also limited in scope and the required matrix multiplications were quite cumbersome. A

prior Design Repository initiative by NIST helped to guide the Design Repository project at Missouri S&T to a more mature state. To enhance data integrity, design information was migrated from spreadsheet form to a relational database. A web-based repository navigator including search and design tool generation features was created along with a repository entry application.

More recently, Missouri S&T has further partnered with UT-Austin [10, 11], Penn State [12], Virginia Tech, Bucknell [13], University of Buffalo and Texas A&M to expand the types of design information and breadth of design tool features within the repository. The Design Repository serves as a hub for designers for information exchange and design generation tools and is heavily utilized in the current VOICE project. Information entry and retrieval occurs within a standalone application [14] (available at http://designengineeringlab.org/repositoryEntry/) while information retrieval occurs over the Internet through the Design Repository's web portal (http://repository.designengineeringlab.org/). The infrastructure supporting these two applications is the Design Repository database and schema [12]. The database schema establishes what types of design information can be stored, the relationship of those elements and the extensibility of including new and additional types of design information.

1.4.5 Concept Generation Techniques

A variety of concept generation methods exist for application to engineering design problems – from those that are common practice within the field of design to the more modern computer aided concept generation methods. Many researchers have sought to formalize the conceptual design phase. Antonsson and Cagan concisely define the notion of 'formal' as "...computable, structured, and rigorous, not ad hoc" [15]. Furthermore, by founding concept generation techniques on functionality, solution-independent design descriptions can be built [16]. Such methods generally rely on a form of functional decomposition of the overall problem to initiate the search for

physical design solutions during conceptual design. Whether driven in this function-based manner or otherwise, much variability is exhibited in just how this search is carried out depending on the method chosen. This reflects the variety of perspectives that have been suggested for addressing the conceptual design problem and a sampling of the major themes is reviewed next.

1.4.5.1 C-Sketch/6-3-5 Method

The 6-3-5 method is a generic technique that supports innovative thinking [17]. In 6-3-5, members of an engineering design team (optimally 6-8 members) generate, interpret, and modify the individual ideas of other team members by first brainstorming and sketching individually on three ideas for various aspects of the product, then passing their ideas to the next team-member who adds additional ideas and sketches. C-Sketch is a variant of the 6-3-5 method wherein members produce only sketches and refrain from communicating verbally when passing ideas to the next member [18]. Passing only sketches allows other team members the opportunity to interpret the concepts in a different manner than the original author, thereby increasing design diversity.

1.4.5.2 The Catalog Design Method

Another approach, referred to as catalog design, is based on a catalog of physical elements (components, assemblies, etc.) that can be browsed for solutions that match required performance specifications. The data for design catalogs are limited to some degree insofar as these design catalogs are generally a subset of previously designed systems, which leads to the issue of potential novelty restrictions. However, a major benefit of catalog design is the ability to utilize design knowledge that falls outside human memory [19-21]

1.4.5.3 Design by Analogy

In Design by Analogy, a functional model is created of the product being designed. Examining analogous products or components that perform the same function generates solutions to the present design problem. The designer then evaluates

these similar components for appropriateness in solving the given design problem [22]. One Design by Analogy method widely recognized in the engineering design community is the Theory of Inventive Problem Solving, or simply TRIZ. TRIZ was developed by Altshuller during the 1940-50's period and was based on the examination of large numbers of existing patents [23]. The end result of this effort is an engineering design approach that identifies a set of conflicts that occur in design along with a set of principles that can be applied to generate solutions that solve these conflicts.

1.4.5.4 Morphological Matrix Method

The morphological matrix introduced by Zwicky is now a classic technique for use in conceptual design [24]. This method provides the design engineer with a simple, albeit manual, means for bookkeeping potential physical solutions and their corresponding functionality.

1.4.6 Foundations in Automated Concept Generation

The front end of the conceptual design process has seen few attempts at automation, perhaps due in part to the evolving strategies and methodologies that exist for this phase of design. However, over the past decade, several methodologies have coalesced around the functional decomposition and partial solution manipulation techniques originally introduced by Pahl and Beitz [25], e.g., [26-35]. These methodologies take a designer through a set of steps to help decompose a design problem and build conceptual solutions based on the functionality that a product needs to exhibit. Function modeling methods abstract the functionality that a solution must fulfill from the established customer needs, ideally removing designer biases that may be introduced by focusing on specific solutions too early in the design process. This abstraction helps a designer generate more complete conceptual solutions and balance design choices between different components with the same functionality [25].

Research into the benefits of structured design methods (e.g., [36]) coupled with research into designers' reluctance to use them (e.g., [37, 38]) seem to point toward the

need for the seemingly tedious stages of systematic design to employ some level of automation to help integrate the benefits of a structured method with the more natural activities of a designer – a need that is most evident during the early phases of conceptual development.

Computational tools for conceptual design do exist, yet these tools often address areas that support aspects such as initial requirements gathering (e.g., organizational tools such as the TikiWiki project [39], the creation of function structures (e.g., the function grammar tool developed by Sridharan and Campbell [40]), or optimization of well-established concepts (e.g., [41]) rather than the translation of functional requirements into creative solutions).

1.4.7 The State of the Art in Automated Concept Generation

Computerized concept generation techniques, spanning the broad AI topics of knowledge representation and reasoning, promise engineers a faster realization of potential design solutions based upon previously known products and implementations. While the area of automated concept generation has made great strides in recent years, most methods still require the user to indicate desired functionality. Using functional descriptions has been shown to help engineers stray away from pre-trained ideas of how a product or device would look and operate, although can cause confusion for engineers and scientists who have not been trained to describe product functionality. Two of the automated concept generation methods under development today rely solely on the user's ability to develop functional descriptions of their desired product. Both of these methods make use of a repository of design information including component connection information and component functionality.

The recent foundations for concept generation through computational reasoning have been developed based on formalisms for describing function or purpose in engineering design largely led by members of our research team [42, 43]. Some of the results of this research include the development of a design repository to allow

designers to store and retrieve design knowledge at various levels of abstraction, from form (components, sub-assemblies and assemblies) to configurations to function. Offering a fully functional and intuitive way to record product design information has been key to the acceptance of repositories as an important concept generation tool for designers. A prototype design repository framework by NIST guided the design repository (discussed further in Section 2.4) project to a more mature state.

The bank of empirical knowledge relating components to functions leads to the development of relational matrices [10, 11] and graph grammar rules [44, 45] that, when combined with a search mechanism, automatically creates conceptual designs. Aiding the methods set forth by Bryant and Kurtoglu [46, 47] is a component naming taxonomy spanning 140 different component classifications. With the open-endedness or large degree of variability in conceptual design, numerous solutions are created through the search mechanisms (on the order of thousands). Presenting these thousands of solutions to the user is similar to an Internet search that produces thousands of results. It is overwhelming to the user and impractical to expect that such a large number of alternatives will be useful to the designer. Furthermore, the results showed that subtle challenges in a given design problem may not always be captured in the specification of initial function, and thus many results were not relevant to the user's needs [48, 49]. As a result, the proof of concept Designer Preference Modeler [50, 51] was created to find within the large set of results which concepts were most meaningful to the designer. By ranking select concepts, the search mechanism learns what aspects of the concept the user prefers, and seeks solutions that maximize the predicted preference. Initial results for this method are promising, but the impact they have on the design process is still unclear.

1.5 HOW TO USE THIS DISSERTATION

The remainder of this dissertation consists of five publications spanning the scope of the engineering design problem outlined above. Combined together the included papers provide tools to aid in the conceptual level of the engineering design process. A repository system is implemented to capture design information regarding the final description of a design and use that information to clarify a project and/or generate concepts. The first paper, "Introduction of a Data Schema to Support a Design Repository," specifically outlines the underlying database and relationships necessary to support product design information archival. With a framework for information archival established, the second paper, "An Open Source Application for Archiving Product Design Information," details an application used for recording product design information. Specifically this paper details information archival from the standpoint of a user and discusses the implementations of an operational repository system. The remaining three papers utilize the operational repository and the data stored within to develop methods that aid in the conceptual stage of design.

The third paper, "Using a Design Repository to Drive Concept Generation," illustrates the first concept generation tool built using data stored in the design repository. Concept generation is simple and takes the form of a morphological function based search that returns possible solution components. Repository data is utilized once again in, "A Natural Language to Component Term Methodology: Towards a Form Based Concept Generation Tool," to develop a list of natural language terms to augment an existing component basis taxonomy. Finally, "Form Follows Form – Is a New Paradigm Needed?" again utilizes repository data and natural language component terms to introduce a form-based conceptual design tool.

1. INTRODUCTION OF A DATA SCHEMA: TO SUPPORT A DESIGN REPOSITORY

Matt R. Bohm,
and **Robert B. Stone, Ph.D.**
Design Engineering Laboratory
Department of Interdisciplinary Engineering
University of Missouri – Rolla
Rolla, MO 65409

Timothy W. Simpson, Ph.D.
and **Elizabeth D. Steva**
Engineering Design Optimization Group
Department of Mechanical & Nuclear Engineering
The Pennsylvania State University
University Park, PA 16802

ABSTRACT

This paper presents the data schema required to capture fundamental elements of design information in a heterogeneous repository supporting design reuse. Design information captured by the repository can be divided into seven main categories of artifact-, function-, failure-, physical-, performance-, sensory- and media-related information types. Each of the seven types of design information is described in detail. The repository schema is specific to a relational database system driving the implemented Design Repository; however, the types of design information recorded are applicable to any implementation of a design repository. The aim of this paper is to fully describe the data schema such that it could be recreated or specialized for industrial or research applications. The result is a complete description of fundamental design knowledge to support design reuse and a data schema specification. The data schema has been vetted with the implemented Design Repository that contains design information for over 100 consumer electro-mechanical products

KEYWORDS: Design Repository schema, conceptual design

1 INTRODUCTION

The objective of a design repository is to allow designers to store and retrieve design knowledge at various levels of abstraction – from form (components, sub-assemblies and assemblies as well as historical performance and failure data) to architecture to function. The different levels of abstraction and types of design information provide innovative ways to approach design. A design by analogy approach, for example, uses a functional or product architecture description to find other existing products which are similar to it, thus providing a starting point for a form solution. A risk conscious approach, for example, uses conceptual level functionality to find related failure information to determine values for risk likelihood and consequence. With a well populated repository, emerging concept generator algorithms take, as input, basic product functionality and synthesize, filter and rank concepts to use as baselines for further product development. While the possibilities design repositories offer are diverse and helpful to designers, the implementation of such repositories are crucial to their overall success and usefulness.

Realizing the potential impact of an operational design repository, researchers at UMR, the University of Texas at Austin and the National Institute of Standards and Technology (NIST) began gathering artifact information in 1999 [1]. Since that time, the process in which artifact data is gathered and recorded has changed significantly. Initially, artifact design information was recorded in spreadsheets and mainly took the form of Bills of Materials (BOM), Function Component Matrices (FCM), and Design Structure Matrices (DSM). While this type of information was useful, it was also limited in scope and the required manipulations to compute with the data were cumbersome. A prior Design Repository Project initiative by NIST helped to guide the design repository project hosted at UMR to a more mature state. To enhance data integrity, design information was migrated from various independent file formats to a relational database. A web-based repository navigator including search and design tool

generation features was created along with a repository entry application creating the user interface for the new repository, dubbed the UMR Design Repository (also referred to as simply the Design Repository in this article).

More recently, UMR has further partnered with UT-Austin [2, 3], Penn State, Virginia Tech and Bucknell [4] to expand the types of design information and breadth of design tool features within the repository. Currently, the Design Repository serves as a hub for designers for information exchange and design generation tools. Information entry and retrieval occurs within a standalone application (available at http://function.basiceng.umr.edu/repositoryEntry) while information retrieval occurs over the Internet through the Design Repository's web portal (http://function.basiceng.umr.edu/repository). The infrastructure supporting these two applications is the Design Repository database and more specifically the database schema. The database schema establishes what types of design information can be stored, the relationship of those elements and the extensibility of including new and additional types of design information.

The objective of this paper is to fully describe the database schema, currently at version 2.0, powering the repository. This paper reports on research efforts to 1) identify pieces of fundamental design information that support designer activities, 2) segment and classify the pieces of design information, 3) define relationships between the disparate pieces of design information, 4) develop ways to standardize design information representation, and 5) deliver a functional database schema. Sections 3 and 4 provide implementation level details, Section 5 presents the larger context of repository operations and Section 6 summarizes conclusions prior to future work in Section 7.

2 BACKGROUND

Several types of applications have been created to record pieces of product or process information. This niche of design-based applications includes the NIST

Repository initiative, PDM (Product Data Management) systems, and CAD based repositories. The NIST Repository initiative set forth guidelines for categorizing function-centric design information. PDM systems allow for part hierarchy storage as well as process data and project management elements. CAD based design repositories store numerous artifact CAD files and rely on feature descriptions and recognition capabilities. In this section an overview of the NIST Repository Initiative, PDM, and CAD based storing systems is presented.

2.1 NIST Repository Initiative

The most similar system to UMR's repository schema takes form in the NIST Design Repository representation model [5-10]. Through their repository initiative, NIST set out to define basic guidelines of a Design Repository and how archived design information could be useful to designers. The NIST Design Repository representation model is a basic framework to help guide what type of product information is collected and how the elements of information are related to each other. NIST has also developed a mapping from this representational framework into an XML (eXtensible Markup Language) data format. While portions of the NIST initiative overlaps with design representation standards such as STEP (Standard for the Exchange of Product model data), the breadth and scope of implementation differ greatly. Like certain STEP protocols, the NIST framework provides for geometric and process information storage but also expands them to a higher-level domain of design information storage.

The NIST initiative proposes a set of information models to be used for modeling product knowledge at varying levels of detail. There are several data entities which allow for a variety of aspects of a product description to be represented. The classes specified in the NIST Core Product Model include: Artifact, Function, Transfer Function, Flow, Form, Geometry, Material, Behavior, Specification, Configuration, Relationship, Requirement, Reference and Constraint.

While the UMR schema contains elements similar to that of the NIST schema but is distinguished by allowing design information regarding customer needs, component basis designations, manufacturer, failure modes and sensory level information to be stored. Also, the NIST schema is only exemplary and has not been implemented in a distributable, publicly accessible and operational system.

2.2 PDM Systems

In recent years product data management (PDM) systems have emerged to help store and retrieve product and part data. PDM systems allow for part hierarchy storage as well as process data and project management elements. Svensson and Malmqvist [11] explore a PDM system and demonstrate many uses of such a system. The PDM system demonstrated collects requirement, function, concept and part structures as well as property models. Additionally a PDM system stores the entire product structure, variants, revisions and finally documentation and CAD models. Although function structures and property models can be stored within a PDM system, they are not capable of storing the detailed function based information we desire and integrating it into useful design tools without heavy modification. A PDM system is a highly effective tool for use in the manufacturing side of emerging products and parts but is fundamentally different from a repository system. Within the UMR repository, similar pieces of design knowledge, such as CAD models and part hierarchies, are stored; however, the main focus is the mapping between functions and components and the compatibility of components to connect together as a system

2.3 CAD-based Systems

Regli [12] in partnership with NIST and the National Science Foundation (NSF) has also developed a CAD-based design repository. The focus of Regli's work includes collaboration in the field of CAD, engineering design, manufacturing process planning, and feature recognition. The design repository contains mostly CAD, solid models, and assemblies along with some supporting documentation such as cost and assembly plans.

3 UMR DESIGN REPOSITORY CONVENTIONS

The UMR Design Repository is an artifact-centric repository, meaning that for a design attribute to exist, it must be linked to an artifact. Other information classes contained in the Design Repository, such as manufacturing and physical parameters for example, describe additional design attributes while still relating to their artifact hub. Because the Design Repository is artifact-centric, understanding the artifact table and associated relationships is key to understanding the data handling capabilities of the repository.

The repository schema is built and served by a PostgreSQL (a SQL variant) database [13]. In general, the database contains tables that have clusters of similar types of information. Database tables are then connected to other tables within the database to form data relationships. In this paper there are two types of database tables: 1) a database table description, which detail the fields and associated data types that define a database table (alternatively, for non-computer scientists, table descriptions describe the structure and connectedness of the data), and 2) a database table that details the set of data entries in the database (alternatively, the actual data that is entered into the repository – in this case, the product data). Note, every database table has an associated database table description. The term row will be used to describe an entry in a database table and the term field will be used to describe an object in the database table description. Table 1 shows the artifact table description in the repository database schema.

All table descriptions throughout this paper are presented in the same format as Table 1. The first column represents the field name, the second column specifies the data type, the third column denotes whether or not a piece of information is mandatory, and the fourth column describes any default values, if applicable. The fifth column describes the type of key that might exist for a particular field. Only one primary key can exist per table and is used to develop a unique reference to the particular entry in that database

table. Having a foreign key designation means that the particular field references the primary key of another database table. All database tables in the UMR Design Repository begin with an id column that is a serial integer, mandatory for any information entry with no default value and designated as the database table's primary key.

Table 1. Artifact table description

artifact				
field name	*data type*	*mandatory*	*default value*	*key type*
id	serial	yes	N/A	primary
name	varchar	yes	N/A	N/A
child_of_artifact	int	no	N/A	foreign
basis_name	int	no	N/A	foreign
serial_id_number	varchar	no	N/A	N/A
assembly	boolean	yes	FALSE	N/A
description	varchar	no	N/A	N/A
quantity	int	yes	1	N/A
system	int	yes	N/A	foreign
manufacturer	varchar	no	N/A	N/A
trademark	varchar	no	N/A	N/A
artifact_release_date	date	no	N/A	N/A
entry_date	date	yes	N/A	N/A
modification_date	date	no	N/A	N/A
creator_info	int	yes	N/A	foreign

Within the UMR Design Repository, there are two main categories of tables—those that store artifact-specific design data information and those that store taxonomies and bases to classify design information. The Design Repository makes use of several taxonomies and bases to describe information such as functionality, failure modes, manufacturing processes, materials and color. While several different taxonomies exists to describe these types of information, ones chosen for use within the repository could alternatively be replaced for specific repository implementations. The tables that store taxonomies and bases are denoted with _type after the table name. Figure 1 shows all 41

of the repository database tables with the 13 database storing tables highlighted. Taxonomy and basis storing tables do not reference design data storing tables; however, they may reference themselves in order to establish hierarchies. Shown in Table 2 is a prime example of a basis-storing database table: the subfunction_type table description.

As with all other database table descriptions in the repository, the subfunction_type table begins with a serial id that establishes a primary key. The second field of the database table is where the actual Functional Basis term is stored. Tier is used to denote whether the particular Functional Basis term is in the primary, secondary, or tertiary level [14]. Child of subfunction establishes a hierarchy of the Functional Basis terms and the definition field is used to hold the definition of the particular function.

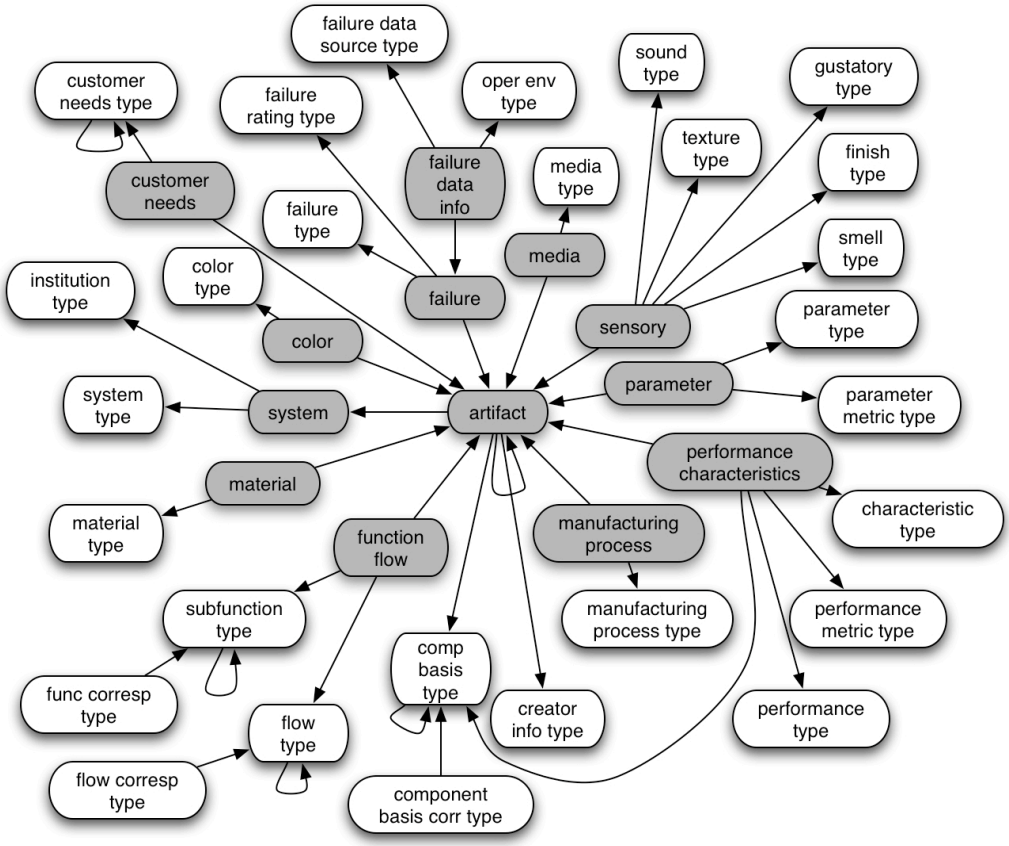


Figure 1. Graphical view of repository database tables

Table 2. Subfunction_type table description

subfunction_type				
field name	*data type*	*mandatory*	*default value*	*key type*
id	serial	yes	N/A	primary
subfunction	varchar	yes	N/A	foreign
tier	int	yes	N/A	N/A
child_of_subfunction	int	no	N/A	foreign
definition	varchar	no	N/A	N/A

4 UMR DESIGN REPOSITORY DATA GROUPS

The design information captured by the Design Repository data schema can be broken up into seven main classes: artifact-, function-, failure-, physical-, performance-, sensory- and media-related information types. All seven of these categories are represented in different database tables but are all brought together by the use of database table relationships, found in the database table descriptions for each database table. In this section, each of the seven data classes are reviewed along with the specific pieces of information they hold. Section 5 details how these elements are connected together to create a cohesive Design Repository.

4.1 Artifact-related Design Knowledge

As mentioned in Section 3, the artifact table serves as a central hub for the remaining six categories of data. Although all design information typically references an artifact, there are a few pieces of design information that the artifact database table stores directly. Each artifact comprises a row entry in the artifact table. An artifact can be considered an entire product, a sub-assembly, or a single part when stored in the Design Repository. To represent the artifacts of a product in the repository, the product is first identified as an artifact, and then all individual assemblies, sub-assemblies, and artifacts are grouped accordingly under that artifact. The repository database has the capability of establishing parent-child relationships such that a product artifact hierarchy is created. In order to keep a strict separation of different products within the

repository a system database table is used, the system database table description is shown in Table 3.

Table 3. System table description

system				
field name	*data type*	*mandatory*	*default value*	*key type*
id	serial	yes	N/A	primary
name	varchar	yes	N/A	N/A
system_type	int	yes	N/A	foreign
description	varchar	no	N/A	N/A
contributing_institution	varchar	yes	N/A	N/A

Looking back at the artifact database table description shown in Section 3 (see Table 1), there is a placeholder for a system reference for each artifact instance. A unique system id is established for each new product that is added to the repository. Every artifact belonging to the given system is then referenced to the system id. In the system database table description (see Table 3), a system name, system description, and contributing institution are associated with the system. For example there may be 30 artifacts named 'motor' that are unique to different products because of the system designation. The contributing_institution field in the system database table is used to track what institutions have recorded design information for a particular product. The system database table also includes a system_type field. The system_type field links to the system_type database table containing a list of different product categories. Example product categories include consumer, industrial, commercial, automotive, space, etc.

The artifact database table description (see Table 1) begins with a serial-based id number to establish a unique serial number for each artifact that resides in the database. Moving through the artifact table, data fields such as the artifact name, description, quantity, manufacturer, trademark, artifact release date, entry date and modification

date are present. The child_of_artifact field is used to create an artifact hierarchy; this is accomplished by designating the field as a foreign key, which in this instance points to an artifact id.

Next in the artifact database table, the basis_name field is used to associate a component basis name to a specific artifact [15]. For example an artifact denoted as a coffee cup would reference the component_basis_type table to establish that 'reservoir' is the corresponding component basis term. Component basis naming is used to cluster similar artifacts. When an artifact is a grouping of several artifacts, the assembly field is used. The assembly field Boolean value defaults to FALSE indicating that the artifact is a singular artifact. For bookkeeping purposes, the creator_info field is used. The creator_info field references the creator_info_type database table, which contains contributor information such as their name, email address, and affiliation.

4.2 Function-related Design Knowledge

Product functionality is highly important not only to conceptual design but also to other design and optimization methods that use function as a link to existing design information. Since several aspects of design engineering and product design revolve around function, it is highly necessary to accurately represent artifact functionality digitally.

The function_flow database table in the repository is used to allow portions of functional models to be associated with an artifact. In order to accurately capture the material, energy, and signal flow through a product, it is necessary to have additional artifact connection information alongside the standard function and flow language. Capturing the function, flow, and artifact connection information is done by associating an input and output artifact and flow with each function. Table 4 shows the function_flow database table description.

Table 4. Function flow table description

function_flow				
field name	*data type*	*mandatory*	*default value*	*key type*
id	serial	yes	N/A	primary
describes_artifact	int	yes	N/A	foreign
supporting	boolean	yes	FALSE	N/A
input_artifact	int	yes	N/A	foreign
input_flow	int	yes	N/A	foreign
subfunction	int	yes	N/A	foreign
output_flow	int	yes	N/A	foreign
output_artifact	int	yes	N/A	foreign

Similar to the artifact database table, the function_flow database table begins with a serial id number creating a unique primary key. The primary key ensures that each set of function and flow descriptions are represented uniquely in the scope of the entire set of function-flow descriptions in the repository. Each tuple containing the {input_artifact, input_flow, subfunction, output_flow, and output_artifact} is linked to a specific artifact by the describes_artifact field. The supporting field is used to establish whether a particular function tuple is described as a supporting or conceptual function [16]. A conceptual function is a function that is required by customer needs where supporting functions describe the necessary functions required for the physical embodiment of the product. The supporting field also has a default value of FALSE, which corresponds to a function being recorded as a conceptual function. The input_artifact and output_artifact fields are both foreign keys that reference a specific artifact id number in the artifact table. The subfunction field is also a foreign key and references a specific function id in the function_type table. All of the data elements in the function_flow table are specified as mandatory in order to accurately represent functionality. In cases where an artifact solves multiple functions the input and output artifact fields can be designated as 'internal,' representing that a particular flow stays within an artifact's boundary. If an input (or output) flow comes from (or goes to) more

than a single artifact, the flow can be designated as going to (or from) multiple sources using the 'internal' designation. For example, when an artifact has an incoming flow of electrical energy from two specific sources both electrical energy flows would be transferred to the designated artifact with an output_artifact of 'internal.' Functionality of the artifact can then be recorded using 'internal' as the input flow. When multiple artifacts are used in concert to solve a single function each artifact is denoted with the overall function. All fields within the function_flow database table are set as mandatory. From a functional perspective, it would not make sense to list a function without also listing the incoming and outgoing flows or the destination.

Table 5 shows the function_flow database table populated with sample data to demonstrate how function relationships are generated. Reading across the table, the sample function and flow tuples describe artifact number 0000008. In the row beginning with an id of 1, the input_artifact corresponds to 'external' and the output_artifact corresponds to 0000009. If an input or output artifact is denoted as 'external' it means that a particular source or destination of a flow crosses the given product's boundary. Both input_flow and output_flow reference id numbers in the flow_type table. For this example, flow id of 16 corresponds to 'electrical energy.' The subfunction field in Table 5 references an id number in the subfunction_type table, with an id of 12 representing the function 'import.' All of these designations for row id 1 correspond to 'electrical energy' being imported from an outside source with a destination of an artifact having an id of 0000009.

Moving on to row id 2 of Table 5, the artifact being described has an id number of 0000009, an input artifact id of 0000008, input flow of id 16, subfunction id of 22, output flow id of 44, and a destination artifact id of 0000006. Translating the id numbers, the row reads as having an input flow of 'electrical energy,' the subfunction 'convert' and an output flow of 'rotational mechanical energy.' Adding both of these rows together shows that two separate artifacts are being described: one that would take form as an

electric plug or wire (artifact id 0000008) and the other artifact taking form as some kind of electric motor (artifact id 0000009). The input artifact for the electric cord is external while the output artifact is the motor. The electric motor has a source artifact of the electric cord while the destination artifact, specified as artifact id 0000006, would likely be some kind of coupler, gear, or other artifact that can connect to an electric motor. A translated version of Table 5 is shown in Table 6. For both Tables 5 and 6, the functions are described as conceptual functions, taking the value of FALSE in the supporting field [16].

Table 5. Function flow database table with sample data

function_flow							
id	*describes*	*input_artifact*	*input_flow*	*subfunction*	*output_flow*	*output_artifact*	*supporting*
1	OOOOOO8	external	16	12	16	OOOOOO9	FALSE
2	OOOOOO9	OOOOOO8	16	22	44	OOOOOO6	FALSE
3	xx	xx	xx	xx	xx	xx	xx
4	xx	xx	xx	xx	xx	xx	xx
5	xx	xx	xx	xx	xx	xx	xx

Table 6. Function flow database table with translated sample

function_flow							
id	*describes*	*input_artifact*	*input_flow*	*subfunction*	*output_flow*	*output_artifact*	*supporting*
1	electric cord	external	electrical energy	import	electrical energy	electric motor	FALSE
2	electric motor	electric cord	electrical energy	convert	rotational mechanical energy	OOOOOO6	FALSE
3	xx	xx	xx	xx	xx	xx	xx
4	xx	xx	xx	xx	xx	xx	xx
5	xx	xx	xx	xx	xx	xx	xx

4.3 Failure-related Design Knowledge

Failure information in this Design Repository is driven by efforts including the Function-Failure Design Method (FFDM) [17, 18], Risk-in-Early Design (RED) [19], and adaptations of modern Failure Modes and Effects Analysis (FMEA) [20] techniques. FFDM and RED are similar in purpose to generic FMEA methods but strive to provide risk and possible failure information at the conceptual level of design based solely on product functionality. It is necessary to build an infrastructure such that designers and engineers can archive and easily access this critical information. A failure mode taxonomy for mechanical and electrical components has been developed at UMR and is used as the reference taxonomy in this work [17, 18].

Like the function_flow database table description, the failure database table description shown in Table 7, begins with a serial id number and link to the particular artifact being described (describes_artifact). It is necessary that the serial id and describes_artifact fields are present to establish a unique identifier for a given set of failure information and to properly link the failure information to a specific artifact. Next, the particular type of failure is recorded in the failure field. Again, the failure field actually references the failure taxonomy, meaning that only the failure id number is actually entered in failure table.

The next two fields in the failure database table are used to specify the severity and whether the failure mode is an actual or potential failure mode. Typically a 1-5 scale is used to denote severity; however, the failure database table allows a float value to be entered in the severity field. The float value is allowed because not all data contributions are rated on the same 1-5 severity scale. It is necessary to specify whether a particular failure mode is an actual failure mode or is only noted that it 'could' happen. Actual failure modes are those that have been recorded historically where potential failure modes are those that are believed to be physically possible. Because of this very distinct difference it is necessary to record the correct information. The potential field in

the failure data table is a Boolean and has a default value of FALSE indicating that a failure mode is an actual failure mode. In cases where a failure mode is denoted as an actual failure mode, it is necessary to record the number of occurrences, the sample size, and rating type.

The default rating scale assumed in the repository is the 1-5 severity scale [20]. In cases where an alternate failure rating scale is used, the rating_type field in the failure table can be used to reference the rating_type table. The rating_type table can be populated with a list of failure mode rating types, a description of the rating system, and conversion values to the repository standard 1-5 rating scale. When occurrence and sample size data is not available and only failure rate data is specified, the rate field is used to input a float value of the failure rate. Ideally it is better to have occurrence and sample size failure data such that similar artifacts and functions can be clustered to present statistically valid failure likelihood and severity information.

Table 7. Failure table description

Failure				
field name	*data type*	*mandatory*	*default value*	*key type*
id	serial	yes	N/A	primary
describes_artifact	int	yes	N/A	foreign
failure	int	yes	N/A	foreign
severity	float	no	N/A	N/A
potential	boolean	yes	FALSE	N/A
occurences	int	no	N/A	N/A
rating_type	int	yes	1	foreign
sample_size	int	no	N/A	N/A
rate	float	no	N/A	N/A

When more accurate failure mode information is available through warranty or problem reporting databases for example, the failure_data_info database table description, shown in Table 8, can be used to record the additional information. The

failure_data_info database table is used to supplement information in the failure database table by adding the data source, a report number for bookkeeping, the operational environment of the artifact at time of failure, date of incident, and a description. Detailed failure information like this is highly important where human life is a factor in the operation of a device. Unlike the failure data table, which references artifact id numbers, the failure_data_info table references a particular failure id number. Using this referencing scheme means additional failure_data_info information can only be associated with an existing failure mode entry.

The driving force behind the failure_data_info comes from NASA, industry partners, and other academic institutions [21-23]. For safety and or economically critical subsystems it is necessary to accurately record not only the failure modes but also additional descriptions of the failure mode. The data_source field in the failure_data_info table references a data_source_type table. Data source types may take on the form of corporate-specific failure databases, warranty data, or NASA's Problem Failure Reporting (PFR) database [24].

Table 8. Failure database info table description

failure_data_info				
field name	*data type*	*mandatory*	*default value*	*key type*
id	serial	yes	N/A	primary
describes_failure	int	yes	N/A	foreign
data_source	int	yes	N/A	foreign
report_number	varchar	no	N/A	N/A
oper_env	int	no	N/A	foreign
date_of_incident	date	no	N/A	N/A
description	varchar	no	N/A	N/A

4.4 Physical-related Design Knowledge

There are several types of physical-related design information elements that can be used to describe artifacts. This type of information originates from the form aspect of an artifact and can be used to search for components that meet certain manufacturing, material or size criteria. Currently the repository records four main categories of physical design information elements: manufacturing information, artifact material, rough geometric bounding dimensions, and color.

The manufacturing_process database table description, shown in Table 9, is used to denote specific manufacturing processes utilized in the manufacture of the referenced artifact. Similar to most other database tables in the repository, the manufacturing_process database table begins with a serial id number and a reference to a specific artifact id (describes_artifact). The manufac_process_type field in the manufacturing_process database table references an id of a specific type of manufacturing in the manufacturing_type table. Examples of manufacturing types include casting, machining, injection molding, etc. The manufacturing_process database table is used only to link a specific artifact to a type of manufacturing process; process data types are stored only in the manufacturing_type table. It is not required to specify a manufacturing type when recording artifact information; however the repository can record multiple manufacturing processes for each artifact.

Table 9. Manufacturing process table description

manufacturing_process				
field name	*data type*	*mandatory*	*default value*	*key type*
id	serial	yes	N/A	primary
describes_artifact	int	yes	N/A	foreign
manufac_process_type	int	yes	N/A	foreign

The material database table description, shown in Table 10, operates in the same manner as the manufacturing_process database table but is used to link an artifact to material types instead of manufacturing types. Examples of material types in the material_type database table include ABS plastic, aluminum, stainless steel, etc. Like the manufacturing_process database table, it is not required to specify material information when recording artifact data although multiple material types can be specified for each artifact.

Table 10. Material table description

Material				
field name	*data type*	*mandatory*	*default value*	*key type*
id	serial	Yes	N/A	primary
describes_artifact	int	Yes	N/A	foreign
material	int	yes	N/A	foreign

The color database table is similar to the manufacturing_process and material database table descriptions and is shown in Table 11. Each instance of a color association is tracked by the serial id field and is then associated to a specific artifact id by the describes_artifact field. The color field in the color database table references a color id in the color_type table. Like the manufacturing_process and material database tables, multiple colors can be associated with a single artifact.

The parameter database table description, see Table 12, is slightly more complex than the prior physical-related design information tables in the repository. The complexity stems from the table's ability to record several different types of information that are quantifiable aspects of the physical artifact. Fields in the parameter database table begin with a serial id and a reference to a specific artifact id number. Additional fields in the table are parameter_type, parameter_metric_type, and parameter_value.

The parameter_value field is where an actual numerical value for a specific parameter is recorded. When specifying a parameter value it is not only mandatory to specify the type of parameter but also the metric used to measure the specific parameter.

Table 11. Color table description

color				
field name	*data type*	*mandatory*	*default value*	*key type*
id	serial	yes	N/A	primary
describes_artifact	int	yes	N/A	foreign
color	int	yes	N/A	foreign

The parameter_type and parameter_metric_type fields both reference specific database tables. A parameter type is typically classified as a measurement, which includes descriptors of length, width, height, diameter, etc. The parameter table is also used to store artifact cost, where the type of 'cost' is denoted. Once a specific parameter type is recorded it is necessary to also record the associated parameter type metric. For the examples of measurement the metric may include inches, feet, centimeters, etc. Examples of the cost parameter metric type include US dollars, Canadian dollars, Japanese Yen, etc.

Table 12. Parameter table description

parameter				
field name	*data type*	*mandatory*	*default value*	*key type*
id	serial	yes	N/A	primary
describes_artifact	int	yes	N/A	foreign
parameter_type	int	yes	N/A	foreign
parameter_metric_type	int	yes	N/A	foreign
parameter_value	float	yes	N/A	N/A

4.5 Performance-related Design Knowledge

Performance-related design knowledge is captured with 'high level' characteristics that describe the overall functionality of the entire product. These characteristics extend beyond the component level representation of functionality, helping translate product functionality into measurable quantities. As defined in Table 13, designers first specify the type of performance (e.g., power) and then the metrics that are used to define it (e.g., Watts). These characteristics can be inherent to the product or be associated with a specific input or output. If the performance characteristic is tied to a particular component within the product, then the designer can specify this association using the component basis [15]. The value of the characteristic must also be specified.

The same unique primary key and artifact id referencing begin the performance_characterstics database table description, and the performance_type, performance_metric_type, and characterstic_type fields all reference their named type database tables. Specific taxonomies are stored in the type database tables. The component_basis_type field is used to reference the component basis taxonomy when the performance of a specific component of a more generic artifact is described. For example, when a motor is entered (the higher level artifact) but a torque rating is given for the output shaft, the component basis specification should be used so that the characteristic is related to the motor but tied specifically to the output shaft. Continuing this example, the electrical requirements for the motor could specify voltage and current, which would also be linked to the wire connectors for the motor.

In addition to performance characteristics, customer needs are categorized as shown in Table 14. Although customer needs do not always match the performance characteristics one-to-one, it is important to denote the desired performance or function when information is available. Elements of the customer_needs database table description resemble the failure database table description in that occurrences, sample_size, and rate are specified fields. Importance is also recorded in the

customer_needs database table. When importance, occurrence, sample size and/or rate are specified, it is possible to evaluate product functionality and performance versus customer needs specifications [25, 26]. The customer_need_type table allows for a list of unique customer needs to be established and is referenced by the customer_need field.

Table 13. Performance characteristics table description

performance_characteristics				
field name	*data type*	*mandatory*	*default value*	*key type*
id	serial	yes	N/A	primary
describes_artifact	int	yes	N/A	foreign
performance_type	int	yes	N/A	foreign
performance_metric_type	int	yes	N/A	foreign
characteristic_type	int	no	N/A	foreign
component_basis_type	int	no	N/A	foreign
performance_value	float	yes	N/A	N/A

Table 14. Customer needs table description

customer_needs				
field name	*data type*	*mandatory*	*default value*	*key type*
id	serial	yes	N/A	primary
describes_artifact	int	yes	N/A	foreign
customer_need	int	yes	N/A	foreign
importance	int	no	N/A	N/A
occurences	int	no	N/A	N/A
sample_size	int	no	N/A	N/A
rate	float	no	N/A	N/A

4.6 Sensory-related Design Knowledge

The sensory database table captures additional product data related to the five senses as shown in Table 15. Finish defines the visual sheen or luster that covers the largest area of the product and is typically one of three options: brilliant, glossy, or dull. Finish relates additional sight data beyond data organized into color and material. Meanwhile, texture relates to touch and specifies the feel of the product when held. It

can be one of three options—smooth, rough, or coarse—based on the extent to which the customer interacts with the product through touch. A product's dormant and operational smell can be specified using options such as strong, mild, or none.

While smell may not be a concern for many products, some designers pay close attention to the smell of their products such as in the automotive industry when interior smell is an important aspect of their product offering (e.g., the 'new car smell'). Likewise, a product's taste may be an important characteristic to record (e.g., for an electro-mechanical toothbrush); hence, gustatory (taste) data is included and specified using the four human tastes: salty, sour, sweet, and bitter. Finally, the operational sound of the product can be specified using specific dB ratings or general terms such as loud, quiet, etc. We assume that the dormant (non-operational) sound is quiet; therefore, it is not included in the sensory data table.

Table 15. Sensory table description

sensory				
field name	*data type*	*mandatory*	*default value*	*key type*
id	serial	yes	N/A	primary
describes_artifact	int	yes	N/A	foreign
finish	int	no	N/A	foreign
texture	int	no	N/A	foreign
dormant_smell	int	no	N/A	foreign
operational_smell	int	no	N/A	foreign
gustatorial	int	no	N/A	foreign
operational_sound	int	no	N/A	foreign

4.7 Media-related Design Information

There are several types of media that can be associated with artifacts. Media types can take the form of pictures, graphical functional models, graphical assembly models, 2D-CAD files, 3D-CAD files, stereo lithographic (.stl) files for rapid prototyping machines, and many others. Note that all media types are stored as large objects (files)

within the database, thus any associated metadata is also stored. All of the types of media, mentioned and unmentioned, reside in the media table of the repository, the table description of which is shown in Table 16. Instances of media are unique and associated with an artifact, which is demonstrated by the id and describes_artifact field.

Although a single media database table is used to hold all types of media for all of the artifacts, the media_type field allows for specification of the exact media type. The media_type field references an id number in the media_type table. Examples of media types in the media_type table include .jpg, .gif, .stl, .dxf, and .pdf. Having a type associated with a specific piece of media directs the repository software components how to handle and display a piece of media. In cases of .jpg and .gif the repository web site will simply display the image. For .stl, .dxf, and .pdf, the repository web site shows a link for file download and viewing in another application.

Table 16. Media table description

media				
field name	*data type*	*mandatory*	*default value*	*key type*
id	serial	yes	N/A	primary
describes_artifact	int	yes	N/A	foreign
media_type	int	yes	N/A	foreign
data	large object	yes	N/A	N/A

5 DISCUSSION

As discussed in the previous sections, the Design Repository consists of numerous data tables to store design information and relationships. The operational Design Repository contains 41 data tables. The 41 data tables do not include tables that are used to control user access, system authentication and other bookkeeping information. All of the operational design tables and their top level referencing is described and shown in this section. Along with the high-level review of the data

schema, features that are enabled by the particular data schema implementation are briefly discussed.

5.1 A High Level Look at Database Tables

To begin this section, the database tables are broken up into two separate categories: (1) those that directly store product information and (2) those that are referenced by product storing database tables. There are 13 database tables that directly store product information and 28 supporting tables. All of the 13 product information storing tables were discussed throughout Section 4.

Figure 2 shows a snippet of the Design Repository schema (Figure 1) with sample data. The boxes represent data tables and arrows represent data relationships. The sample data shown represents only a small subset of design information that can be associated with an artifact. For this example a "gear" is shown as an artifact along with associated media, functionality and failure information. The arrows that connect the function_flow, failure, and media tables to the artifact table establish a relationship to the artifact "electric motor." The corresponding _type table relationships are also shown. For example, looking at the failure table, the failure is denoted as "28" and corresponds to an entry in the failure_type table as being "high cycle fatigue."

All 41 of the repository data tables are represented in Figure 1 with the 13 data storing tables highlighted. A data table makes a reference to another table by an outbound arrow to a particular data table. Looking at Figure 1, the failure table references the artifact, failure_type and failure_rating_type tables but is referenced by the failure_data_info table. Tables 18 and 19 show a textual version of Figure 1. The product information storage tables are listed in Table 17 while the product information support tables are listed in Table 18. While most of these tables were referenced in Section 4, some of them were not directly discussed. All of the tables listed in Table 17 are referenced by the data tables listed in Table 18; however, some of the supporting tables are referenced by other supporting tables.

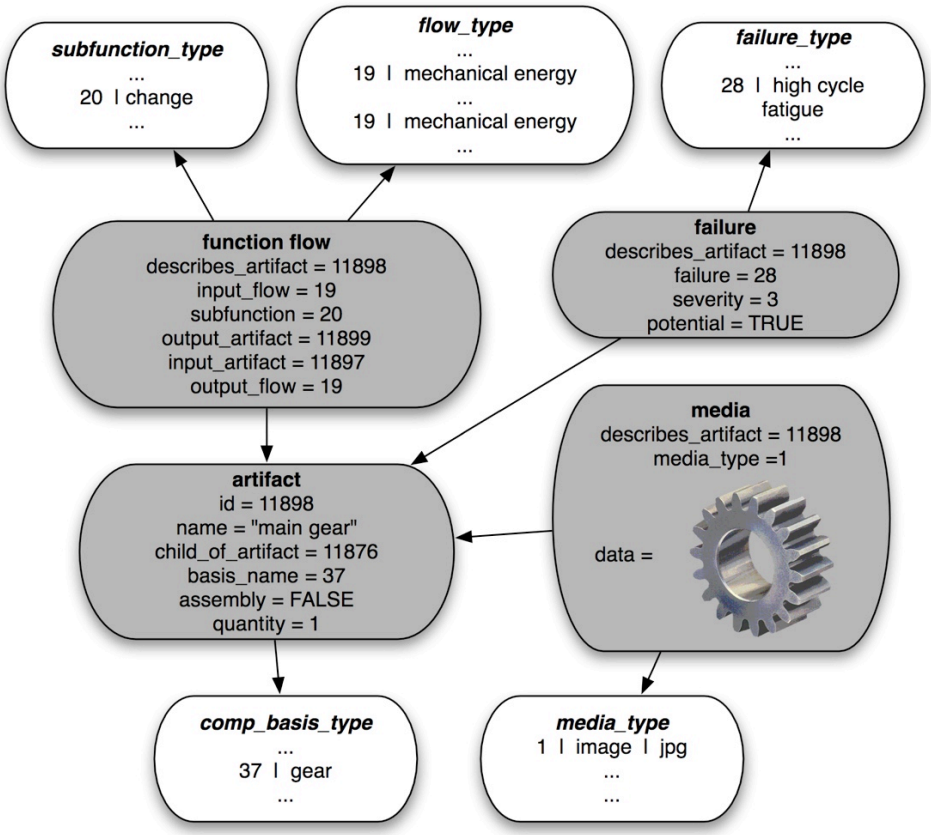


Figure 2. Repository schema snippet with sample data

Table 17. Database table listing

	Database Table Name	Description	References
1	artifact	Used to record high level artifact information such as name, description, quantity	system id, creator info type id, comp basis type id, artifact id
2	color	Used to record the color of an artifact	artifact id, color type id
3	customer needs	Used to record customer needs, importance and the number of occurrences	artifact id, customer needs type id
4	failure	Used to record artifact failure modes, severity and likelihood	artifact id, failure type id, failure rating id
5	failure data info	Used to record additional artifact failure information such as data source and the operational environment	failure id, failure data source type id, oper env type id
6	function flow	Used to record artifact functionality	artifact id, subfunction type id, flow type id
7	manufacturing process	Used to record manufacturing process associated with an artifact	artifact id, manufacturing process type id
8	material	Used to record the material of an artifact	artifact id, material type id
9	media	Used to store media such as photos, cad drawings and functional models of artifacts	artifact id, media type id
10	parameter	Used to record physical measurements and artifact cost	artifact id, parameter type id, parameter metric type id
11	performance characteristics	Used to record performance data such as voltage requirement and output torque	artifact id, performance type id, performance metric type id, characteristic type id, comp basis type
12	sensory	Used to record items relating to the five sense such as sound, sight, etc.	comp basis type id, texture type id, smell type id, finish type id, sound type id, gustatory type id, artifact id
13	system	Used to establish a unique product in the repository	system type id, institution type id

Table 18. Database type table listing

	Database Table Name	Description	Referenced By
1	characteristic type	Used to store the type of performance characteristic (input, output, inherent)	performance characteristics
2	color type	Used to store a list of colors	color
3	comp basis type	Used to store the taxonomy of general components	artifact, performance characteristics, component basis corr type
4	component basis corr type	Used to store synonyms to component basis terms	
5	creator info type	Used to store information about an individual who creates a set of product information	artifact
6	customer needs type	Used to store a list of typical customer needs	customer needs
7	failure data source type	Used to store the list of failure data sources	failure data info
8	failure rating type	Used to store the list of different failure rating scales and conversion factors	failure
9	failure type	Used to store the electrical and mechanical failure taxonomies	failure
10	finish type	Used to store a list of possible artifact finishes	sensory
11	flow corresp type	Used to store synonyms of the flow words of the functional basis	flow type
12	flow type	Used to store the flow words of the functional basis	
13	func corresp type	Used to store synonyms of the function words of the functional basis	
14	gustatory type	Used to store a list of possible artifact tastes	sensory
15	institution type	Used to store a list of types of institutions (academic, industry, etc.)	system
16	manufacturing process type	Used to store a list of manufacturing processes	manufacturing process
17	material type	Used to store a list of material types	material
18	media type	Used to store a list of possible types of media associations and their required actions	media
19	oper env type	Used to store a list of possible artifact operating environments	failure data info

Table 18. Database type table listing (cont.)

20	parameter metric type	Used to store metrics associated with physical artifact parameters (feet, inches, etc.)	parameter
21	parameter type	Used to store a list of possible types of physical parameters (length, width, etc.)	parameter
22	performance metric type	Used to store units for possible performance parameters (dBa, ft-lbs etc.)	performance characteristics
23	performance type	Used to store a list of possible types of performance parameters (torque, power, etc.)	performance characteristics
24	smell type	Used to store a list of possible types of artifact odors	sensory
25	sound type	Used to store a list of possible artifact sounds	sensory
26	subfunction type	Used to store the function words of the functional basis	function flow
27	system type	Used to store a list of possible system types (consumer, industrial, etc.)	system
28	texture type	Used to store a list of possible artifact textures	sensory

5.2 Engineering Design Applications Enabled by the Repository Schema

The Design Repository serves as a hub for several engineering design methods and applications. Emerging techniques in concept generation and preliminary risk assessment are two design methods that both utilize the design repository. Currently the concept generator takes as input a list of desired functions. The concept generator algorithm then queries the database to find existing components known to solve the desired functions. As a second step the concept generator again queries the repository to then determine which of those components are known to physically connect to one another. The final output is a set of concepts that could further be ranked on specific design criteria.

Preliminary risk assessment is an effortless FMEA done at the conceptual level of design where only desired product/device functionality is known. Like the concept generator application, the risk assessment tool also takes as input a list of desired

functions. The repository is then quarried to return information regarding failure modes and failure mode severity. Through some simple matrix based calculations the risk assessment tool then returns a Risk Fever Chart that maps functions and failure modes on a 0-5 likelihood and severity axes. Both of these tools require very little information from the user and quickly return design relevant information.

6 CONCLUSIONS

The UMR Design Repository represents several years of development and has undergone multiple revisions and updates. The current version of the repository schema, web portal and entry application demonstrate that design information can be archived and provide useful tools for designers. Although the task of building and expanding systems to digitally represent design information will continue, the schema presented in this paper provides a roadmap to future revisions and supports design information storage and the associated repository connected applications. The UMR Design Repository—and more specifically the types of design information recorded—began with a somewhat limited set of design information: component-to-component connects, function-to-component connections, and basic artifact bills of materials. The UMR schema has expanded on the initial data set to include failure and risk-based information, generic component naming, function and flow hierarchies, multiple types of media associations, and sensory-related artifact descriptions – potentially a fundamental set of design knowledge upon which future uses will be built.

7 FUTURE WORK

Future work of the UMR Design Repository includes allowing additional data types within the schema, increasing database accessibility and viewing options, and developing additional synthesis and analysis tools. Schema expansion may include data tables and references to enable mathematical analysis, consumer product safety commission reports, and process modeling. Database accessibility can be greatly enhanced by offering a connection API as well migrating the web portal to a Web 2.0

site. Refinements to the existing concept generation and risk analysis tools along with developing easier and more efficient ways to populate repository data will increase repository usefulness.

Schema additions include the integration of mathematical-based transfer functions, function and flow synonym lookup implementation, component naming synonym lookup implementation, further database optimization, work with standardization efforts, and further application development. The desire to include mathematical-based transfer functions stems from work in concept generation. By including mathematical transfer functions, automatically generated concepts can be easily tested against target values. The difficult task associated with including transfer functions is the classification of the associated mathematical formulas and maintaining database integrity when tracing mathematical and numerical variables.

Access and usability of repository information can be increased by providing a repository API to outside organizations. Repository users are sometimes interested in specific subsets data and their relationships. By providing an API users can develop their own customized searches and data views. Upgrading the web portal to a Web 2.0 application would allow for different types of data navigation and interactive design tools to be employed. A Web 2.0 application would also allow the current concept generation and risk analysis tools to be hosted online.

A graphical functional model editor is currently under development at UMR. Further development is necessary to connect the editor to the repository system. The connection will allow an easier way to populate repository data and can also be used as a visualization tool for current repository data.

ACKNOWLEDGEMENTS

This work is supported by the National Science Foundation under Grant Nos. IIS-0325415 and IIS-0325402. Any opinions or findings of this work are those of the

authors and do not necessarily reflect the view of the National Science Foundation or our collaborators.

REFERENCES

[1] Bohm, M., R. Stone and S. Szykman, 2005, "Enhancing Virtual Product Representations for Advanced Design Repository Systems," *ASME Journal of Computer and Information Science in Engineering*, 5 (4): 360-372.

[2] Bryant, C., D. McAdams, R. Stone, T. Kurtoglu and M. Campbell, 2005, "A Computational Technique for Concept Generation," *Proceedings of IDETC/CIE 2005*, DETC2005-85323, Long Beach, CA.

[3] Bryant, C., R. Stone, D. McAdams, T. Kurtoglu and M. Campbell, 2005, "Concept Generation from the Functional Basis of Design," *International Conference on Engineering Design, ICED 05*, Melbourne, Australia.

[4] Shooter, S. B., T. W. Simpson, S. R. T. Kumaray, R. B. Stone and J. P. Terpenny, 2005, "Toward a Multi-Agent Information Infrastructure for Product Family Planning and Mass Customization," *International Journal of Mass Customization*, 1 (1): 134-155.

[5] Szykman, S., R. Sriram and S. Smith, 1996, *Proceedings of the NIST Design Repository Workshop*, National Institute of Standards and Technology, Gaithersburg, MD.

[6] Murdock, J., S. Szykman and R. Sriram, 1997, "An Information Modeling Framework to Support Design Databases and Repositories," *Proceedings of DETC'97*, DETC97/DFM-4373, Sacramento, CA.

[7] Szykman, S., J. Racz and R. Sriram, 1999, "The Representation of Function in Computer-Based Design," *Proceedings of the ASME Design Theory and Methodology Conference*, DETC99/DTM-8742, Las Vegas, NV.

[8] Shooter, S., W. Keirouz, S. Szykman and S. Fenves, 2000, "A Model For Information Flow In Design," *Proceedings of the ASME Design Theory and Methodology Conference*, DETC2000/DTM-14550, Baltimore, MD.

[9] Szykman, S., S. Fenves, S. Shooter and W. Keirouz, 2001, "A Foundation for Interoperability in the Next-Generation Product Development Systems," *Computer-Aided Design*, 33 (7): 545-559.

[10] Szykman, S., 2002, "Architecture and Implementation of a Design Repository System," *Proceedings of DETC2002*, DETC2002/CIE-34463, Montreal, Canada.

[11] Svensson, D. and J. Malmqvist, 2001, "Integration of Requirement Management and Product Data Management Systems," *DETC*, DETC2001/CIE-21246, Pittsburgh, PA.

[12] Regli, W. C. and D. M. Gaines, 1997, "A repository for design, process planning and assembly," *Computer-Aided Design*, 29 (12): 895-905.

[13] Douglas, K. and S. Douglas, 2003, *PostgreSQL*, SAMS Publishing.

[14] Hirtz, J., R. Stone, D. McAdams, S. Szykman and K. Wood, 2002, "A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts," *Research in Engineering Design*, 13 (2): 65-82.

[15] Kurtoglu, T., M. Campbell, C. Bryant, R. Stone and D. McAdams, 2005, "Deriving a Component Basis for Computational Functional Synthesis," *International Conference on Engineering Design, ICED'05*, Melbourne, Australia.

[16] Bohm, M. and R. Stone, 2004, "Representing Product Functionality to Support Reuse: Conceptual and Supporting Functions," *Proceedings of DETC2004*, DETC2004-57693, Salt Lake City, UT.

[17] Stone, R., I. Tumer and M. Stock, 2005, "Comparing Two Levels of Functional Detail for Mapping Historical Failures: You Are Only as Good as Your Knowledge Base," *Research in Engineering Design*, In Press, DOI 10.1007/s00163-005-0005-z.

[18] Stone, R., I. Tumer and M. Van Wie, 2004, "The Function Failure Design Method," *Journal of Mechanical Design*, 127 (3): 397-407.

[19] Grantham Lough, K., R. Stone and I. Tumer, 2005, "Function Based Risk Assessment: Mapping Function to Likelihood," *Proceedings of the ASME International Design Engineering Technical Conference*, Long Beach, CA.

[20] Lock, D., 1993, *Handbook of Engineering Management*, Butterworth-Heinemann, Oxford.

[21] Tumer, I. Y. and R. B. Stone, 2003, "Mapping Function to Failure During High-Risk Component Development," *Research in Engineering Design*, 14 (1): 25-33.

[22] Tumer, I. Y. and R. B. Stone, 2003, "Analytical Methods for Mapping Function to Failure During High-Risk Component Development," *Research in Engineering Design*, 14 (1): 25-33.

[23] Tumer, I. Y., R. B. Stone and D. G. Bell, 2003, "Requirements for a Failure Mode Taxonomy for Use in Conceptual Design," *International Conference on Engineering Design*, Stockholm Sweden.

[24] Tumer, I. Y., R. Stone, R. A. Roberts and A. F. Brown, 2003, "A Function-Based Exploration of JPL's Problem/Failure Reporting Database," *ASME International Mechanical Engineering Congress and Expo*, IMECE2003-42769, Washington, D.C.

[25] McAdams, D., R. B. Stone and K. L. Wood, 1999, "Functional interdependance and product similarity based on customer needs," *Research in Engineering Design*, 11 (1).

[26] Yu, J. S., J. P. Gonzalez-Zugasti and K. N. Otto, 1999, "Product Architecture Based Upon Customer Demand," *ASME Journal of Mechanical Design*, 121 (3): 329-335.

2. AN OPEN SOURCE APPLICATION FOR ARCHIVING PRODUCT DESIGN INFORMATION

Matt R. Bohm, Jayson P. Vucovich and Robert B. Stone, Ph.D.
Design Engineering Lab
Department of Interdisciplinary Engineering
University of Missouri-Rolla
Rolla, MO 65409
mbohm@umr.edu, jayson@vucovich.com, rstone@umr.edu

ABSTRACT

This paper describes an open source computer application developed at the University of Missouri – Rolla (UMR) for archiving product design information. The Repository Entry application is designed to work with the UMR design repository to record and upload product information. Written in C++, the application and user interface is compiled in Qt allowing for native Macintosh and Windows executables. The Repository Entry application can record all of the design information types allowed by the repository including: artifact-, function-, failure-, physical-, performance-, sensory- and media-related information types. By using XML, files can be seamlessly transferred between the Windows and Macintosh entry application versions as well as the online repository. Through an example product the procedure of using the entry application to record and upload design information is demonstrated. The result of this research is a fully functional, easy to use and multi-platform application to aid in the design information archival and reuse process.

1 INTRODUCTION

The objective of a design repository is to allow designers to store and retrieve design knowledge at various levels of abstraction, from form (components, sub-assemblies and assemblies) to architecture description to function. The different levels of abstraction and types of design information provide innovative ways to approach

design. A design by analogy approach, for example, would use a functional or product architecture description to find other existing products, which are similar to it, thus providing a starting point for a form solution. A risk conscious approach, for example, would make use of conceptual level functionality and failure related information to determine values for risk likelihood and consequence. With a well-populated repository, emerging concept generator algorithms can take as input basic product functionality and instantaneously develop, filter and rank concepts to use as baselines for further product development. While the possibilities design repositories offer are diverse and helpful to designers, the implementation of such repositories is crucial to their overall success and usefulness. Offering a fully functional and intuitive way to record product design information is key to the acceptance of repositories as an important tool for designers.

Realizing the potential impact of an operational design repository, researchers at UMR began gathering artifact information in 1999 [1]. Since that time, the process in which artifact data is gathered and recorded has changed significantly. Initially, artifact design information was recorded in spreadsheets and mainly took the form of Bills of Materials (BOM), Function Component Matrices (FCM), and Design Structure Matrices (DSM). While this type of information was very useful, it was also limited in scope and the required matrix multiplications were quite cumbersome. A design repository initiative by the National Institute of Standards and Technology (NIST) helped to guide the design repository project at UMR to a more mature state. The need to have a unified point of entry for design information was initially fulfilled by the EBOM (Enhanced Bill of Materials) entry application [1]. The EBOM entry application was a simple stand-alone database form implemented with FileMaker Pro. Although the EBOM application was functional, it lacked an intuitive interface, and transferring design information to the online repository was a daunting task.

With increasing usage of the UMR design repository as well as the number of outside institutions contributing to the repository database of design information, it

became necessary to enhance the features, reliability and quality of the EBOM application. Of key interest was the way in which repository contributors recorded product information as well as uploaded design information to the central repository database. All previous versions of the EBOM application required that the contributing partner email a FileMaker file to a UMR contact to then be parsed and uploaded to the repository database. Another downfall of the EBOM entry application was the lack of version control. Often times, a contributing partner would record product information using an outdated version of the EBOM application and cause data-mismatches with the repository database. Using all of these items as central customer needs, an entirely new repository entry application was designed and built from the ground up. The new repository entry application can be downloaded at http://function.basiceng.umr.edu/repositoryEntry.

The goal of this paper is to present and fully describe the new repository entry application. This paper reports on research efforts to 1) streamline the process of design information entry, 2) eliminate redundancy in the entry process, 3) implement tools to verify design information integrity, 4) create an easy way to transfer design information and 5) deliver an easy to use repository entry application. Section 3 presents technical details of the repository application including: object classes, XML frameworks and database connectivity. Section 4 demonstrates how product design information is recorded and uploaded to the repository database.

2 BACKGROUND

Several types of applications have been created to record pieces of product or process information. This niche of design-based applications include PDM (Product Data Management) systems, CAD based repositories and knowledge storing systems. PDM systems allow for part hierarchy storage as well as process data and project management elements. CAD based design repositories store numerous artifact CAD files and rely on feature descriptions and recognition capabilities. Knowledge based

applications record information regarding product or artifact function, flow or customer need attributes. In this section an overview of PDM, CAD based and knowledge storing systems that includes a wide variety of design-based information systems is presented.

2.1 PDM Systems

In recent years product data management (PDM) systems have emerged to help store and retrieve product and part data. PDM systems allow for part hierarchy storage as well as process data and project management elements. Svensson and Malmqvist [2] explore a PDM system and demonstrate many uses of such a system. The PDM system demonstrated collects requirement, function, concept and part structures as well as property models. Additionally a PDM system stores the entire product structure, variants, revisions and finally documentation and CAD models. Although function structures and property models can be stored within a PDM system, they are not capable of storing the detailed function based information we desire and integrating it into useful design tools without heavy modification. A PDM system is a highly effective tool for use in the manufacturing side of emerging products and parts but is fundamentally different from a repository system. Within the UMR repository, similar pieces of design knowledge, such as CAD models and develop part hierarchies, are stored; however, the main focus is the mapping between functions and components and the compatibility of components to connect together as a system.

2.2 CAD-based Systems

Regli [3] in partnership with NIST and the National Science Foundation (NSF) has also developed a CAD-based design repository. The focus of Regli's work includes collaboration in the field of CAD, engineering design, manufacturing process planning, and feature recognition. The design repository contains mostly CAD, solid models, and assemblies along with some supporting documentation such as cost and assembly plans.

2.3 Knowledge Based Systems

Several researchers have built a variety of knowledge-based design information systems and have used different product representations with varying degrees of design

knowledge abstraction. Although not all of the knowledge bases built were designed for the collection of function-based design, information can still be extracted from these systems and their representations.

Summers [4] reports on a feature-based knowledge system designed for CAD-based feature elements such as shapes, protrusions or cuts with the intent of supporting designer activities. For conceptual designers, features are crucial pieces of form-based modeling information. The modeling approach of a CAD-based feature system and a functional-based representation are different; however, both capture relevant design information. Design information in both cases is highly relevant to the respective fields of study. What differentiates the two approaches is the application of the design representations. The CAD-based knowledge system from Summers exemplifies that all design knowledge is relevant dependent upon the domain, representation and level of abstraction. Dixon [5] makes the point that feature descriptions and representations must be valid within their respective domain of use.

Functional representations have been used to represent design information in early repository systems. These systems used a block diagram approach and were based on "function logic." One of these early systems, described by Sturges [6] and powered by Hypercard stacks, was used to navigate function diagrams. In this preamble to the Functional Basis and defined functional modeling techniques, a representation schema had to be chosen. The representation schema used by Sturges [6] built on function logic to describe complex systems and included mathematical relationship equations in relationship to the "function blocks." Through the use of "function logic" and "function blocks," designers were able to gain insight into how a product operates functionally

3 ENTRY APPLICATION TECHNICAL DETAILS

In this section the technical underpinnings of the entry application are explored. Section 3.1 details how the entry application is organized and programmed. Section 3.2 explains how the entry application connects and transfers data with the online

repository. Finally, Section 3.3 describes the XML framework used to save repository documents.

3.1 Building the Entry Application

The Repository Entry application is an object oriented data management tool and designed to work with the UMR repository version 2.0 database schema [7]. The interface is designed to guide the user in capturing relevant design information for entry into the repository database. Because this information is taken from tangible items, the object-oriented paradigm is highly suitable for implementing the software. Here, the organization of the software will be discussed, with key features highlighted as appropriate.

Modules in the Repository Entry software are organized into C++ objects or classes. Each object class corresponds to a logical or physical component of the product or the interface used to gather product information. For example, much like the Design Repository, the central component of data is the *Artifact* class. This object represents an actual artifact from the product being represented. The class captures the data prescribed by the Design Repository schema and stores it in an *Artifact* object in memory. This class also gives the object the appropriate operations needed to interface it with other *Artifact* objects, the system to which they belong and the user interface used to capture the data. This object-oriented organization allows for a logical mapping of the software to its requirements and applications.

Following the Design Repository schema, *Artifact* objects are collected into another object class representing a system. *System*s are further collected into an object representing the repository. Various further encapsulations of data are found in classes that can be part of an *Artifact*. These include classes for *Failure*s, *Physical Attribute*s, and of course, classes that represent the artifact's *Function* and *Flow* components.

Each of these objects possess the ability to validate the data entered into them through an interface with the object representing the repository and to provide feedback

and interaction with the user interface. The user interface is constructed using Trolltech's Qt framework [[A]]. This allows for rapid development by leveraging existing objects and customizing them for this particular application. Appropriately, Qt provides software building blocks, each with some core functionality that can be connected and manipulated to build new software, allowing the software engineer to focus on satisfying customer needs rather than re-inventing the wheel.

Qt was chosen as the framework for several reasons. Primarily, Qt allows for immediate cross-platform development. The Qt framework utilizes libraries native to the target system to ensure that an application compiled for Microsoft Windows has the familiar Windows interface, while an application compiled for OS X uses native OS X interface components. At the same time, the Qt framework provides a platform-agnostic application programming interface (API) [[B]]. This removes the burden of maintaining multiple versions of code for multiple target systems.

Secondly, Qt was chosen because it is strongly object-oriented. Objects created from the Qt framework can interact with one another using signals and messages. The framework already includes several useful building blocks that allow for rapid development of applications. Among these included objects are those that provide database connectivity, network and internet connectivity, operating-system agnostic file access, XML document creation and manipulation and, of course, user interface capabilities [8].

Each of the data centric classes described previously has a corresponding user interface class that is used to both capture and display the data found within an object of the given class. In Qt parlance, user interface objects are *Widgets*. The *Artifact Widget* contains the drop-down boxes, text fields and custom interface objects that guide the user in capturing artifact data. Like the *Artifact* class itself, the *Artifact Widget* is a composite of other, more narrowly defined widget classes (*e.g.*, the *Failure Widget*).

These widgets combine and work together to guide the user in defining a product that is composed of various artifacts. The user is prompted to fill out identifying information for an artifact first (such as artifact name, description, component name, etc). Next, the user finds controls for entering functionality performed by the artifact. Functionality relies on *Flow*s coming in from and going out to other artifacts. The interface makes it equally easy to select an input or output artifact from a list of those already instantiated as to direct the application to create a new artifact to be described later from the function/flow entry area.

Since artifact identification and function/flow association are the principle forms of data, the interface widgets that capture this information are displayed foremost in the application. Other attributes are captured from widgets lying on other tabs behind the functionality tab. The user is guided in a similar manner to associate failure data (*i.e.*, failure modes, failure severity, failure consequence, etc), physical parameters (*i.e.*, dimensions and physical measurements), human sensory information (*i.e.*, texture, smell, taste, etc), and many other attributes.

After entering in information describing an artifact, the user can use an intuitive tree structure to organize the artifacts into a *System* or product. This hierarchy is represented by an interactive tree structure. Newly created artifacts are placed in a special area where they can be dragged and dropped into their appropriate position in the system tree. This allows for the representation of the product as a collection of sub-assemblies.

Additionally, the entry application allows the user to provide a photo of the artifact through a simple drag-and-drop interface. Other various files prescribed in the Design Repository schema can also be associated. This information is encapsulated with the artifact and later uploaded into the repository database.

Several checks are in place to ensure the integrity of the data captured. First, data integrity is assured where possible by constraining certain attributes to lists of pre-

defined options (such as failure mode, functionality, color, etc). For these attributes, the set of possible values is finite and known *a priori*. However, to improve the experience of advanced users, many of these fields allow the user to type the value as though it were a free-text entry area. The interface transforms this free text into the closest matching value from the fixed data space. These pre-defined values are packaged in a mutable form with the software, which will be discussed in more detail in Section 3.2.

Secondly, the creation, deletion, or modification of an artifact prompts a quick validation of the product's data integrity. Checks are performed to ensure, among other constraints, that no two artifacts have the same name and that every artifact that is referenced exists within the system. If an error is found, the user is notified through the *Problem Artifact* menu, which lists the offending artifact and provides access to resolve the issue.

Finally, the structure of storage and retrieval provides a mechanism to ensure data integrity. Each system, when exported to a file, is a self-contained unit, making external reference only to a set of "legal" values for each of the captured attributes. This set of terms is fairly stable and derives from the Design Repository schema itself. The Design Engineering Lab maintains the set of terms centrally. The Repository Entry Application contains a mechanism to ease the transition from one version of the set of terms to the next. This translation framework allows a software engineer to simply add an entry into the term mapping dictionary to be incorporated in the next software release (which ostensibly coincides with the release of the new set of terms). When one of the self-contained product data structures is opened, either from a local file or a remote repository, the terms used are translated as necessary into their new counterparts.

3.2 Linking the Entry Application and Online Repository

The Repository Entry application implements two primary means of information transfer. Products can be imported from and exported to self-contained XML files or

through a remote repository database connection. Each mode has a separate interface class providing the functionality.

Remote repository connections are made using Qt classes for SQL-based database interaction. Qt provides a means to execute SQL queries on remote databases and then to retrieve the results. This functionality is incorporated into a custom repository database interface class. This class contains many SQL statements written to bridge the divide between the internal object representation of a product and the remote repository's database table representation of a product.

Because both the Repository Entry Application and the Design Repository database are built using object-oriented design paradigms, it is a relatively straightforward task to map the logical representation of a product from one to the other. Little transformation has to occur to move a product from the repository database to the Entry Application or vice-versa.

A typical interaction between the Entry Application and the Design Repository is described:

1. The user launches the Entry Application and elects to connect to a remote repository using a URL or IP address, username, and password.
2. The Entry Application connects to the remote repository and verifies the username and password and, if valid, retrieves the user's permissions which control whether or not the user is able to download products, upload products, delete or modify products, and/or modify the repository itself.
3. The user is presented with a list of products contained in the repository.
4. The user selects a product to be opened by the Entry Application. Because the Design Repository is a relational database, the database interface object queries the database to gather the attributes associated with the product's artifacts. This information is dynamically retrieved, downloaded, and used to construct the objects described above.

5. The user is presented with the system in a ready-to-edit format with the product now residing entirely in local memory. Further database interaction is not necessary.
6. The user may make changes to the product.
7. The product can be saved locally as a self-contained XML file to be shared with other users of the Repository Entry application or, if the user has the proper credentials, uploaded to the Design Repository, in which case the process outlined in step 4 is executed more or less in reverse: Each internal artifact object is decomposed into primitive data which is inserted into the appropriate database tables using SQL queries.

3.3 XML File Structure

As has been previously mentioned, products can be saved in self-contained XML files. The import and export of products into XML files is handled by an interface class similar to the repository database interface class described previously. This XML interface class is built using Qt XML and DOM (Document Object Model) classes. These classes are used to translate the internal object structure of a product into a text file formatted as XML.

Most of the attributes of an artifact can be represented as simple text strings. This makes using XML as a transmission medium very straightforward. Additionally, XML files can be structured in a hierarchical way, which closely mimics the internal object structure used by the Entry Application, which in turn mimics the hierarchical structure of the Design Repository schema.

An advantage to using XML to store products is that the files are human readable. Opening an XML file created by the Entry Application (which uses the *.repo* file extension) will reveal a structure which could be understood by a user familiar with the Design Repository. Most importantly, these XML files are self-contained. All of the data necessary to reconstruct the product as entered is present in the *.repo* file. These files can easily be e-mailed back and forth or shared across a network. This is

accomplished by encoding the binary data associated with artifacts into a text representation.

The XML interface module is also responsible for loading the set of valid values for many of the constrained attributes captured by the Entry Application. This set of terms was discussed previously and includes such constraints as the Functional Basis, the set of available failure modes, the allowed component basis names, and many other such sets of possible values.

In addition, this set of terms also contains auxiliary information about many of the terms. For example, each function described by the Functional Basis includes a definition of the function, information about the function hierarchy, and a list of synonyms (correspondents) to that function. This information is used to build a searchable reference dictionary for the user as well as to construct the hierarchical presentation of function and flow values in the *Function* and *Flow Widget*s.

This set of terms is stored as XML and packaged with the Repository Entry application in several ways. Primarily, the set of terms is built into the application when it is compiled. This allows the set of terms to be available regardless of remote repository connectivity for stand-alone operation. Secondly, the set of terms can be loaded from a special terms XML file. In this way the user can manually update their set of terms to a newer version (which is then stored either in a registry key for Microsoft Windows users or in a preference file for OS X and Linux users). Thirdly, the set of terms can be ascertained by querying a remote repository database and dynamically constructing the set.

Most importantly, however, the Repository Entry application will check a central location on the Design Engineering Lab server for information on the latest application and set of terms versions. If the set of terms is out of date, the user can choose to download and install the newest set of terms from within the Entry Application itself.

Additionally, if the application itself is out of date, the user will be notified and presented with instructions on how to obtain a new version of the Entry Application.

4 USING THE ENTRY APPLICATION

There are two main ways in which the repository entry application can be utilized. The first way is to download, review and edit existing design information. The second way is to use it as a dissection aid to enter new product information. In this section both usage methods are described with focus on use as a product dissection and design information capture aid. Section 4.1 describes using the repository entry application as a capture aid while section 4.2 describes using the application as a way to download, review and edit existing design information. To use the application it must first be downloaded from http://function.basiceng.umr.edu/repositoryEntry. The entry application can be used as a product dissection and information entry tool without a repository account. In order to use the entry application to download, view and edit existing design information a repository account is required. To activate a repository account, navigate to http://function.basiceng.umr.edu/repository and click on the "Create an Account" tab.

4.1 Recording New Product Information

The repository entry application is an artifact-centric method of storing design information, meaning that for a design attribute to exist, it must be linked to an artifact. Other information classes contained in the design repository entry application, such as manufacturing and physical parameters for example, describe additional design attributes while still relating to their artifact hub. Product information can be recorded simultaneously with or following the decomposition process. Throughout this section, a vise grip is used to show the design information entry process. Prior to entry, the vise grip was decomposed, and a functional model was created shown in Figure 1.

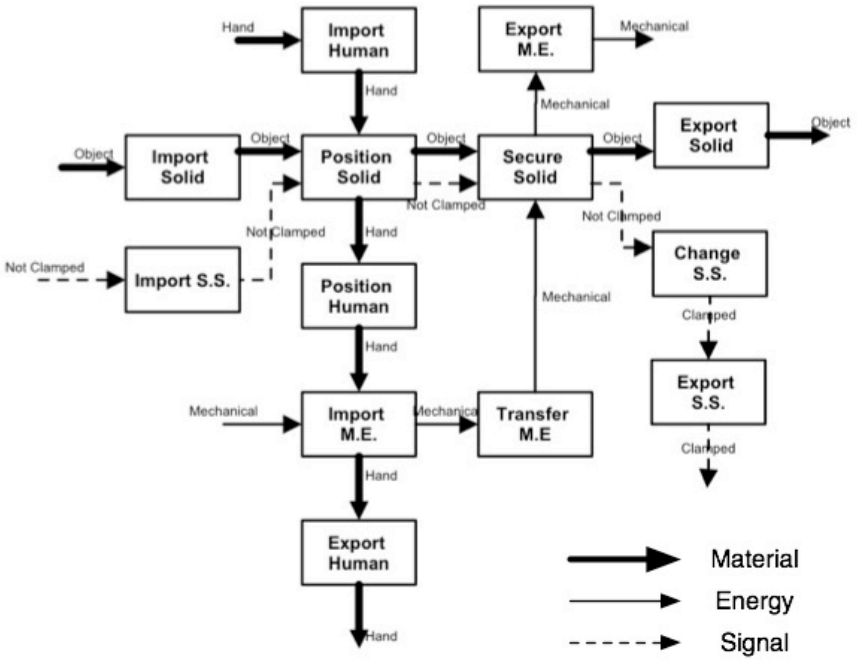


Figure 1. Vice Grip Functional Model

4.1.1 Creating New Artifacts

When first launching the repository entry application the user is presented with a screen asking them to define a new product or connect to the online database (Figure 2). To use the entry application as a means to record a new product, high-level product information is initially requested. High-level product information includes the system (product) name, type of product, description, contributing institution information and general contact information. The available system types (product types) are currently limited to consumer, industrial, NASA and scientific. The system types, like many other pieces of information stored by the repository, draw from various taxonomies that are controlled by the "repository.termsXML" file. Once high-level system information is entered, product information entry can occur after clicking the Create System button.

Once the Create System button is clicked the main entry application window is initialized using the system name as the window title. Within this window, shown in

Figure 3, information regarding artifact function, attributes, failure, files and hierarchy can be entered. On the left side of this screen there are areas for an artifact image, unattached artifacts and an artifact tree. Across the top of this window there are buttons labeled add artifact, create new artifact, clear artifact, view systems and save system. At this point a blank file for the vise grip as a system has been initialized but no design information can be entered until an artifact(s) have been created.

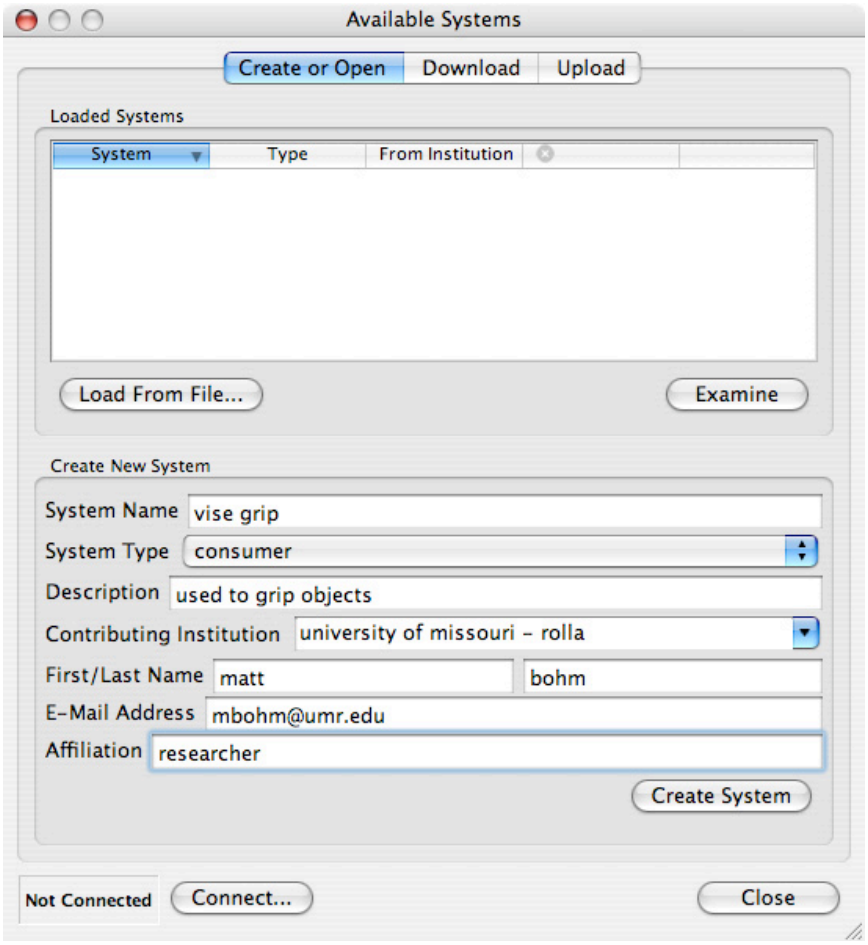


Figure 2. Creating a New System

To create a new artifact, simply click the Create New Artifact button at the top of the window. Notice that once the Create New Artifact button has been clicked areas of

the window that were previously shaded are now unshaded and accessible. There are two main ways in which the entry application can be used to enter new product data. The first way is to create all of the artifacts initially and then proceed to enter artifact information such as functionality. Another way is to create a new artifact and add all of the associated design information before traversing to a new artifact. For large products it usually works best to create all of the artifacts and then enter their associated design information. For smaller products either approach can be used. In this example all of the artifacts will be created before actual design information is entered.

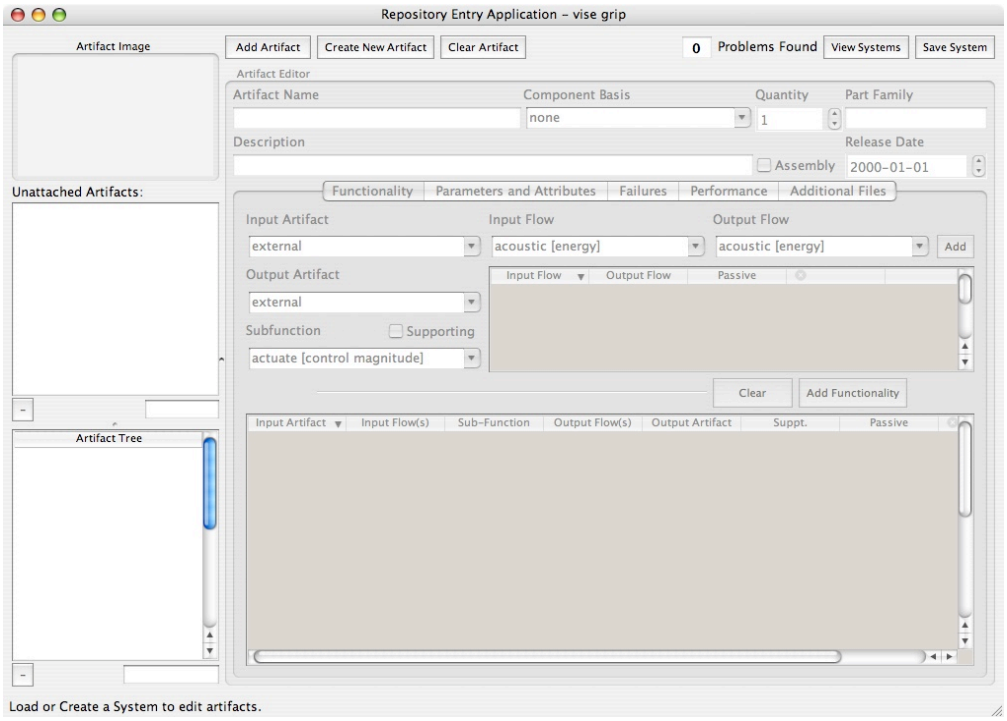


Figure 3. Blank Product Entry Screen

The top-level artifact in the vise grip system is the vise grip itself. With the entry application ready to accept input, the artifact *vise grip* is entered by typing the name in the Artifact Name box and clicking the Add Artifact button. Once the Add Artifact

button is clicked, the artifact *vise grip* now appears in the Unattached Artifacts box in the middle left of the application window. This process is now repeated for all of the artifacts contained within the vise grips. Figure 4 shows the Unattached Artifacts box of the main application window populated with the vise grip artifacts. The artifacts are labeled as unattached because they have not yet been placed within the Artifact Tree. The Artifact Tree is used to establish a hierarchy of artifacts within a product.

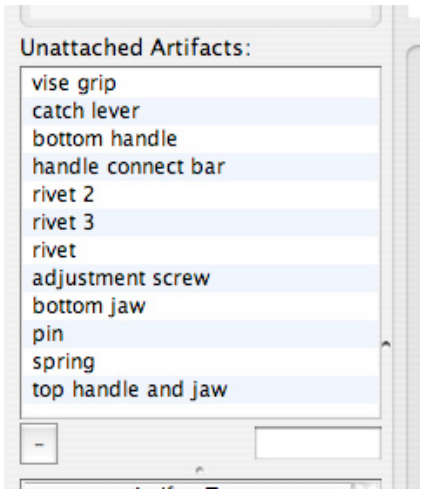


Figure 4. Unattached Artifacts Listing

With all of the artifacts created and present in the Unattached Artifacts box they can now be moved to the Artifact Tree box. During this step of product entry it is important to consider the hierarchy of the product. The vise grips are very simple and do not contain complicated subassemblies therefore all of the artifacts will be children of the main vise grip artifact. To establish this type of relationship the vise grip artifact is dragged to the Artifact Tree box. Since all of the remaining artifacts are children of the main vise grip artifact they must be situated beneath the vise grip artifact. The remaining artifacts are dragged one-by-one from the Unattached Artifacts box and dropped on the vise grip artifact. After dropping the first artifact on the vise grip artifact

an arrow should appear to the left of the vise grip artifact. The arrow denotes that there are now children artifacts of the vise grip artifact. To view the children artifacts simply click on the arrow such that it is pointed downward. Figure 5 shows a populated Artifact Tree box and the hierarchy of the vise grip artifacts.

Now that all of the artifacts are situated within the Artifact Tree, additional artifact information can now be entered. The main categories of artifact information include: functionality, parameters and attributes, failures and media.

4.1.2 Adding General Artifact Information

To begin, a user would want to add general artifact information such as an image, description, component basis name, quantity, release date and whether or not the artifact in question is an assembly. All of these information fields are located in the upper portion of the entry application window. To add any information to an artifact, the artifact must first be selected in either the Unattached Artifact or Artifact Tree boxes. Since all of the artifacts of the vise grip have already been added to the Artifact Tree they will be selected from this point.

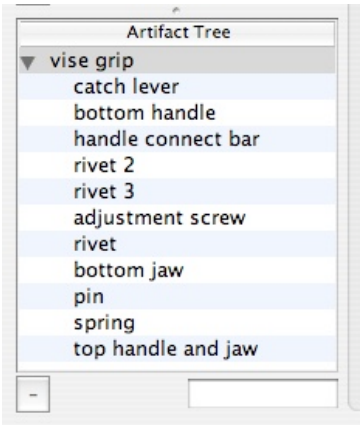


Figure 5. Artifact Tree Listing

To associate an image with an artifact simply drag an image file to the Artifact Image box in the upper left hand corner of the application window. To keep overall file size relatively small it is best if images are jpeg format and are no bigger than 640x480 pixels. Once an image has been dropped on the Artifact Image box a thumbnail of that image will appear. The Component Basis box of the entry application allows for a common component name to be associated with an artifact [9, 10]. Since the vise grip artifact is an assembly of several artifacts and does not exist within the component basis taxonomy the box will be left at its default value of none. The Quantity box allows for the numerical quantity of an item to be entered. If a particular screw existed 18 times within an artifact, it would be captured by the Quantity descriptor. The Part Family field allows for a part family name to be associated with an artifact. This could be used in an industrial setting where there might exist 25 different types of the same artifact. The Description box allows for a free text description of the item. In cases where an artifact is an assembly of several artifacts the Assembly checkbox is used. The vise grip artifact is an assembly of artifacts and will be checked for this artifact. Release Date can be used if the release date of a specific product is known but in most cases is left blank. Figure 6 shows populated general information for the vise grip artifact.

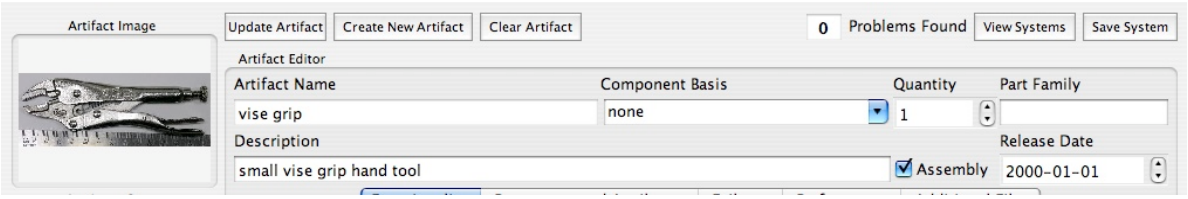


Figure 6. General Information of the Vice Grip Artifact

4.1.3 Adding Artifact Functionality

The center section of the repository entry application allows for information on functionality, geometry, manufacturing processes, materials, failure modes and media

files to be associated with an artifact. A user can access these different types of information by clicking across the Functionality, Parameters and Attributes, Failures and Additional Files tabs in the entry application window. Although a tab is shown for Performance, this section of the entry application remains as future work. To add artifact functionality, click on the Functionality tab of the entry application. Multiple functions that consist of an input artifact, input flow, subfunction, output flow, output artifact and a supporting designation can be associated with each artifact.

Both the input and output artifacts are used to trace flow from the current artifact to the corresponding input and output artifacts. Input and output flows are used to trace the actual flow of the artifact while the subfunction box is where functionality can be described. The entry application makes use of the Functional Basis to describe both function and flow [11]. The supporting function option is used to denote whether the artifact subfunction is supporting or conceptual [12]. Figure 7 shows a close-up of a blank artifact functionality entry screenshot.

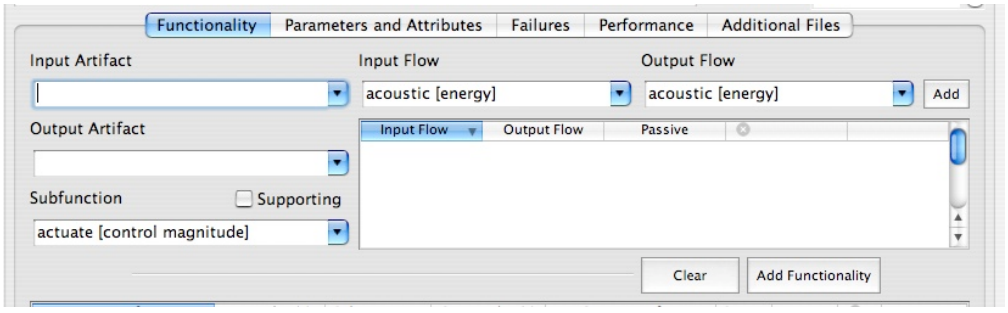


Figure 7. Blank Artifact Functionality

Since the vise grip artifact is the top-level artifact of the product the Black Box functional description will be used. For all other lower-level artifacts the standard functional model will be consulted to add artifact functionality. The Black Box functional description of the vise grip is to secure a solid with flows of mechanical

energy, an object, human material and clamped/not-clamped signal. Since this is at the Black Box level of description all input and output artifacts will be set to external. Lower-level function descriptions will use appropriate input and output artifacts to denote function and flow sources and destinations. To add functionality for the vise grip artifact, begin by entering the input and output artifacts as external and a subfunction of secure. The subfunction secure will be used for all of the flows at the black box level. Next, the input and output flow boxes are set as solid material. To add this as a flow, click the Add button to the right of the Output Flow box. Continuing on with secure as the subfunction, change the input and output flow boxes to mechanical energy and once again click the Add button. Repeat this process for all of the flows that operate on a particular function. Figure 8 shows the flows populated with secure as their function. Notice that Passive checkmarks have been indicated next to the mechanical energy, human material and signal flows. This is because the function secure only truly operates on the flow of solid materials. The flows of energy, human material are simply carrier flows [13]. If you wish to remove a flow, click on the gray circle 'x' next to a flow. Once all of the flows associated with a particular function have been added, click the Add Functionality box to add a function to an artifact.

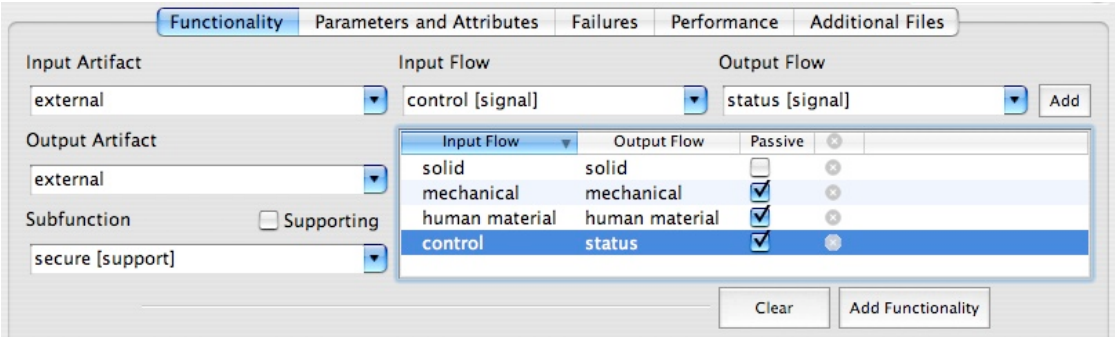


Figure 8. Artifact Function and Flow Entry

The entry application allows for multiple flows to be associated with a single function. Alternatively, each function could also contain only a single active flow. For the case of the vise grip artifact, all of the flows passed through the same function. If only a single flow is associated with a particular function the Add Functionality button would be clicked after recording the particular function-flow combination. From there, additional functions and flows can be associated with an artifact. Figure 9 shows the completed function listing for the vise grip artifact. Notice that the entry application recognizes the function secure as having multiple input and output flows with the same input and output artifacts.

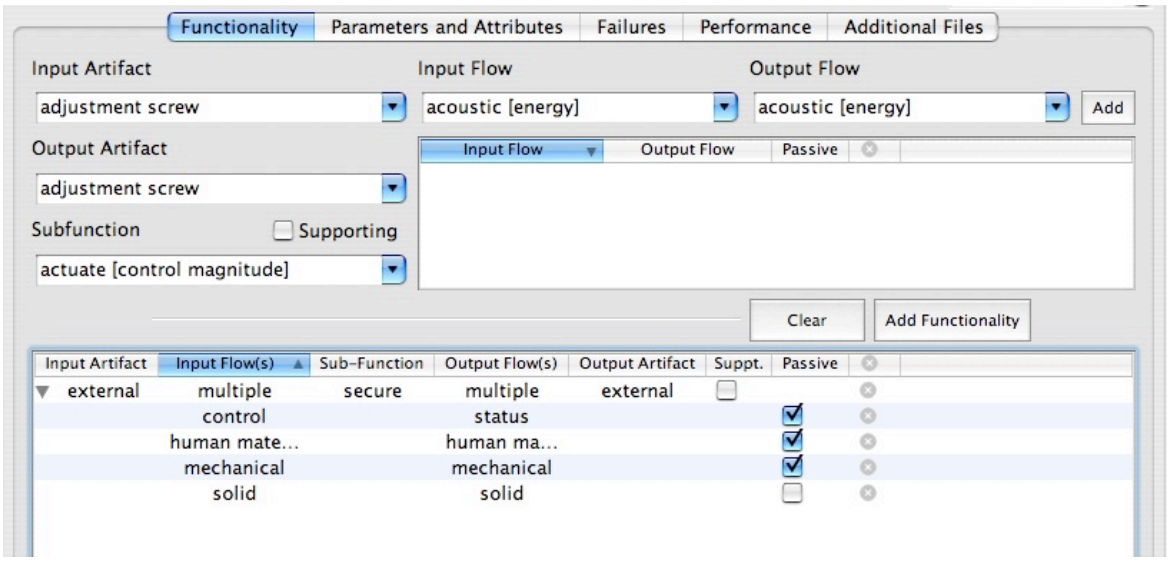


Figure 9. Artifact Function Listing

4.1.4 Adding Artifact Parameters and Attributes

The Parameters and Attributes tab allows materials, manufacturing processes, colors, physical dimensions, finishes and textures to be associated with an artifact. Each subsection within the Parameters and Attributes tab contains a list to select an attribute and a Plus button to add an attribute. For example, to record the vise grip material, use

the Materials pulldown list to select *steel* and then click the Plus button (Figure 10). This same process is used to populate the remaining Parameters and Attributes sections. Since the vise grip artifact is an assembly of several different parts it does not make sense to list manufacturing process. The overall color of the vise grip artifact is listed as gray. The Physical Parameters section is used to describe overall geometries of an artifact. To describe geometries use the Dimension pulldown list to select a specific geometric attribute. With a geometric attribute selected, enter the corresponding numerical value and use the remaining pulldown list to select the appropriate unit. Once the attribute type, value and unit are entered click the Plus button to record the information. The Physical Parameters section can also be used to record artifact weight and mass.

The lower right corner of the Parameters and Attributes tab is used to record design information regarding the five senses: sight, sound, taste, touch and smell. These senses correlate to surface finish, sound, taste, texture and smell. The vise grip artifact can be described as having a glossy surface finish and a smooth texture. Sensory information is recorded in the same manner as for artifact material. Select the sensory descriptor from the appropriate pulldown menu and click the Plus button to record the information. Figure 10 shows populated a populated Parameters and Attributes tab for the vise grip artifact.

4.1.5 Adding Failure Information

The Failure tab (Figure 11) of the entry application allows for actual or potential failure and severity information to be associated with an artifact. Failure information in this section is driven by efforts including the Function-Failure Design Method (FFDM) [14, 15], Risk-in-Early Design (RED) [16], and adaptations of modern Failure Modes and Effects Analysis (FMEA) [17] techniques. FFDM and RED are similar in purpose to generic FMEA methods but strive to provide risk and possible failure information at the conceptual level of design based solely on product functionality. A failure mode

taxonomy for mechanical and electrical components has been developed at UMR and is used as the reference taxonomy here [14, 15]. To associate actual or potential failure modes with an artifact, begin by selecting the appropriate failure mode through the Failure Mode pulldown tab. The Severity box allows for a severity of the failure mode to be recorded. Typically a 1-5 scale is used with 1 being a minor artifact malfunction and 5 being catastrophic and leading to the loss of life. If the failure mode is an actual documented failure mode the Actual Failure checkbox should be enabled. If the failure mode in question is an actual failure mode, the lower portion of the Failure tab can be utilized. Information regarding the number of occurrences, sample size, failure rate and detailed failure reports can be recorded. This specific information can then be used to perform risk analysis. Once all relevant failure information for an artifact is entered the Add Failure button should be clicked to then associate the failure information with the artifact.

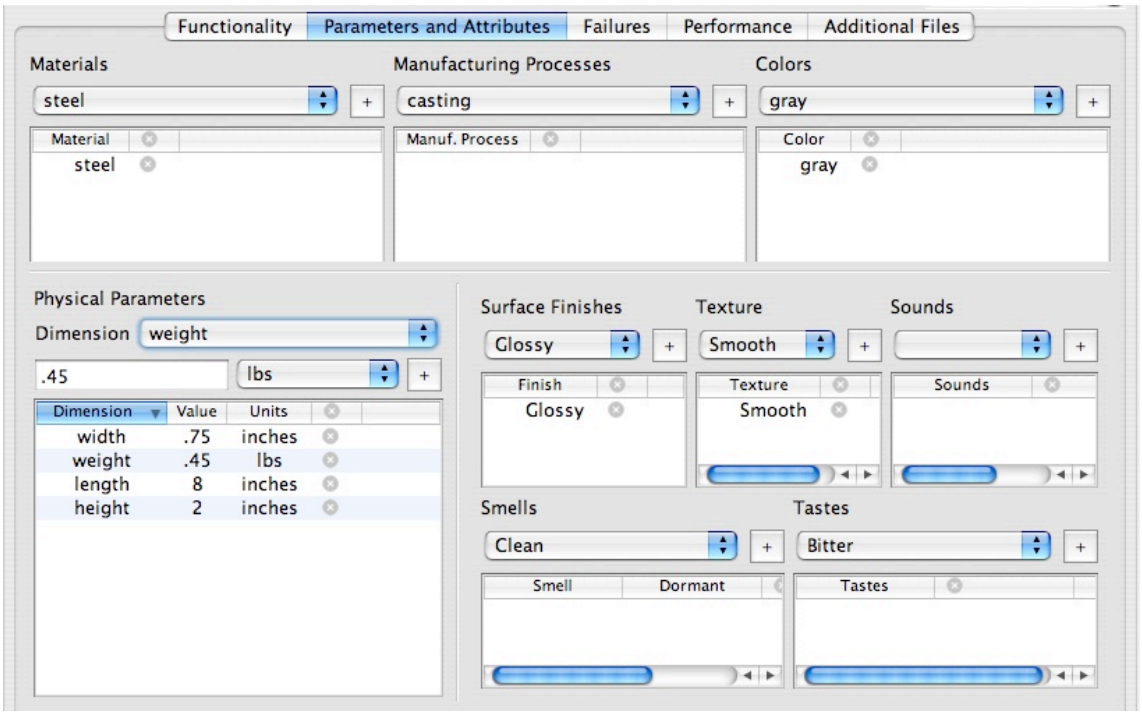


Figure 10. Parameter and Attributes Tab

4.1.6 *Associating Additional Files with an Artifact*

The Additional Files tab (Figure 12) of the entry application allows for assembly model, functional models, CAD files and VRML files to be associated with an artifact (Figure 12). Typically functional and assembly models are associated with the top-level artifact; in this case the vise grip artifact. There are drop boxes for both functional and assembly model PDFs as well as a source file (the actual Concept Draw, Omni Graffle or Visio file). To add a functional or assembly model to an artifact simply drag the appropriate file to the labeled well of the Additional Files tab. CAD and VRML files can also be associated with an artifact and are done so by also dragging the appropriate file to the labeled drop box. Any of the associated files can be retrieved through the entry application by clicking the Save File button. A dialog box will prompt for the name and location for the file to be saved locally.

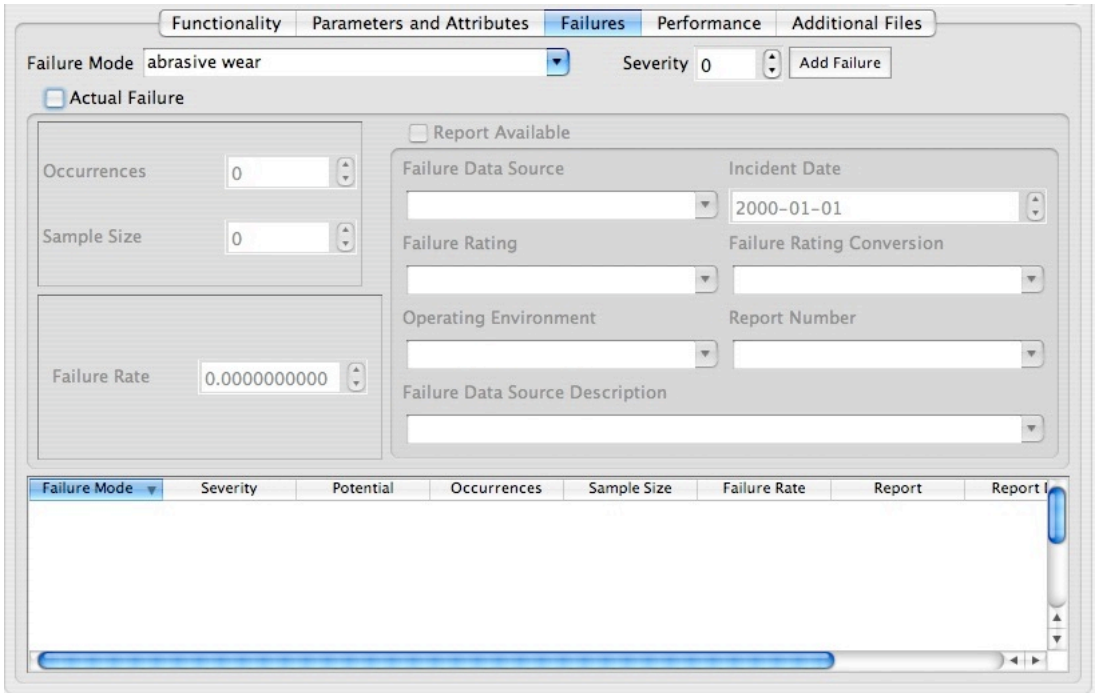


Figure 11. Failure Tab

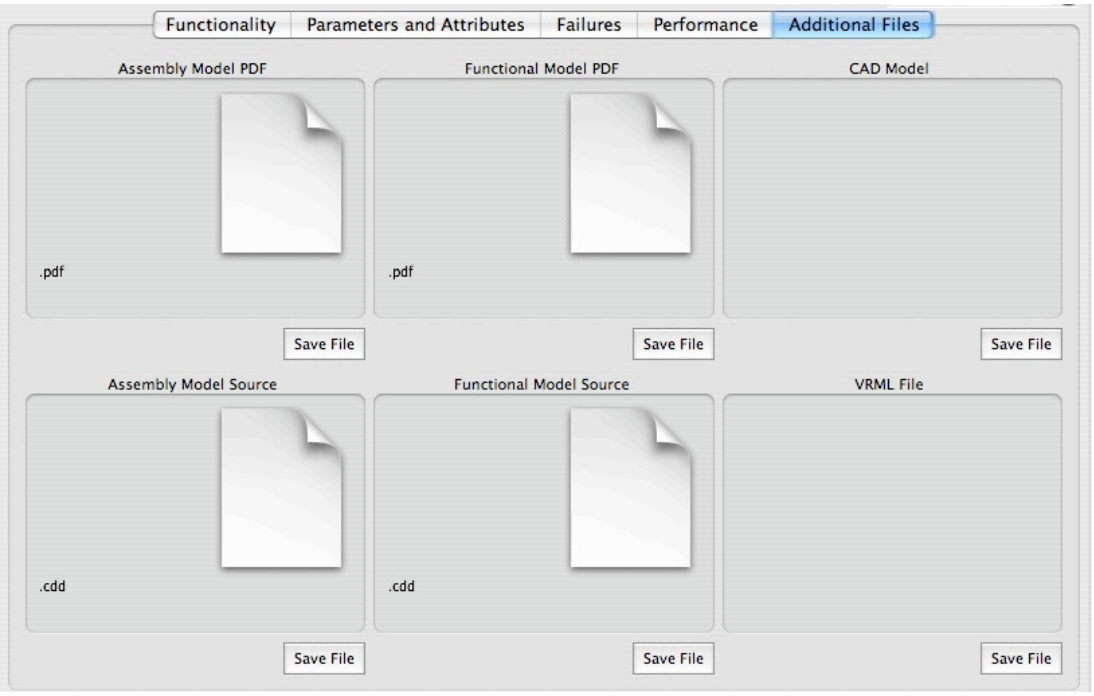


Figure 12. Additional Files Tab

4.1.7 Completing the Product

Once all additional design information as been associated with a particular artifact, click the Update Artifact button in the main entry application window. Doing so will commit all of the changes made to the artifact. To complete a product, follow the same procedure for each artifact contained in the product. To save the product file click the Save System button in the upper right hand corner of the entry application window. The entry application will then ask the user to provide a name and specify a location for the ".repo" file to be saved.

4.2 Using the Entry Application to Open or Edit an Existing Product

The entry application can open or edit exiting product files that are stored locally on a machine or remotely hosted by the UMR repository. The Available Systems window is used to access all existing files. If the Available Systems window is not present it can be activated by going to the Windows tab and clicking Show Systems (Figure 2). To open a local .repo file click the Load From File button under the Create or

Open Tab. Navigate to the appropriate .repo file and click Open. The product can then be seen in the Loaded Systems box. Selecting the file and clicking the Examine button will then open the file in the entry application.

To examine a product that is housed in the repository database, the Download tab of the Available Systems window is utilized. In order to access the online database a user must first be connected to the repository. To connect to the repository click the Connect button at the bottom of the screen (Figure 13). A window titled "Connect to a Repository Database" will then appear on screen. Authentication to the online repository can be gained by using your repository username and password (Section 4). Once the appropriate user name and password have been entered the click the Connect button. The user will then be returned to the Available Systems window showing all of the products from the repository (Figure 14).

To open a repository-based system simply click on the system name and then click the Examine button. Please be patient as the selected system downloads over the Internet and is constructed dynamically from the database. All authenticated repository have access rights to download products, edit them and save them locally. If a user wishes to have repository upload privileges, they can contact the Design Engineering Lab.

The privilege level of the user's account governs repository access. There are four levels of access. The most basic level is *User*. A *User* is able to download remote systems and nothing more. The next level, *Contributor*, adds the ability to upload systems to the repository. The third level, *Administrator*, has the additional ability to mark uploaded systems as verified and to delete systems. Finaly, a database *Superuser* also has the ability to edit fundamental database tables, create, edit, and delete user accounts and grant user access privileges.

Uploaded systems are not immediately available for viewing through the online Design Repository web interface. After uploading, a system must be marked as verified

by a repository administrator. This provides a level of quality control over the data that is presented online. Uploaded systems, regardless of verification, can always be downloaded using the Entry Application.

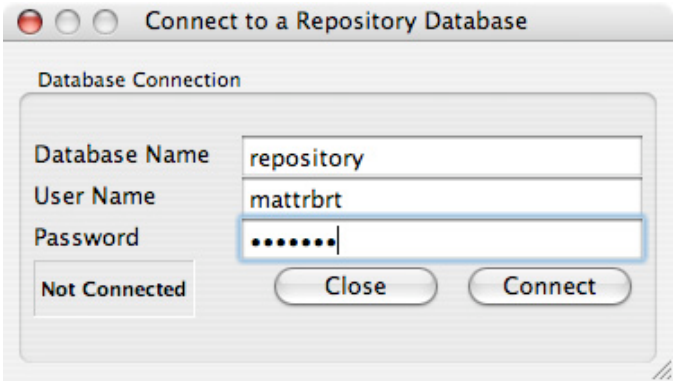


Figure 13. Repository Connection Window

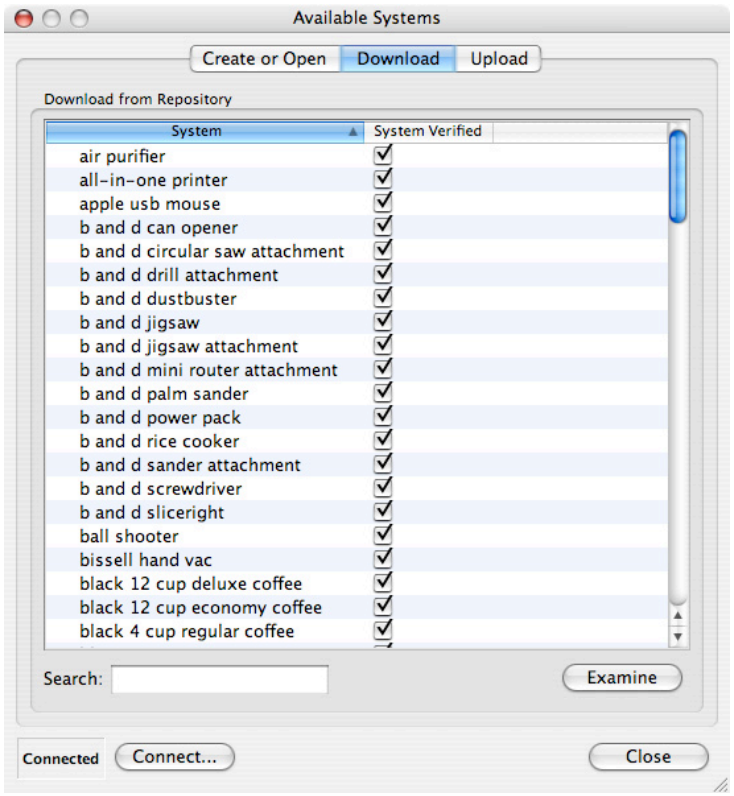


Figure 14. Available Repository Products

5 CONCLUSIONS AND FUTURE WORK

The entry application represents several years of research and development in the field of product dissection, archival and reuse. Although the task of building and expanding systems to digitally represent design information will continue, the entry application presented in this paper provides a roadmap to future revisions and supports design information storage with modern repository applications. The current version of the entry application stable, functional and supports multi-platform development and execution. The entry application is released under the GNU General Public License and as such the source code is available and serves as a baseline for other developers.

Future work of the entry application includes implementing the ability to record mathematical performance equations, usability analysis and further integration into the suite of function-based applications currently under development. Currently neither the repository database schema nor entry application supports the archival of mathematical-based performance equations. Including mathematical transfer functions alongside artifacts would allow for future concept generator tools to analyze overall concepts based on various input and output parameters. A user would also be able to quickly search for particular components based upon values for certain parameters such as input voltage or output torque. The main difficulty in implementing such a feature is the way in which mathematical equations are stored and related across all artifacts. While simple transfer functions and mathematical equations would be easy to implement, those in the form of partial differential equations pose the most difficulty.

In order to fine tune the entry application a series of usability and case studies are required. Through use case and usability analysis, a great deal of information regarding user interaction could be gained. This information could then be used to adjust the cosmetic layout of the entry application and provide for a better user experience.

Currently, an application named Function CAD is being developed at UMR. Function CAD is a program that allows users to draw graphical functional models. While several applications exist that can create a functional model, Function CAD allows users to start from existing functional model snippets to use as building blocks for a larger more detailed functional model. Function CAD also obeys functional modeling rules and will not allow the user to create an inappropriate functional model. For example, a free form drawing package would allow the representation of human energy as a solid material (as they are represented by different line styles); Function CAD would not allow this representation as it is both syntactically and semantically aware of the functional model. Function CAD also uses the same XML backbone as the entry application. Further refinement of both applications would eventually allow a functional model to be drawn in Function CAD and then imported to the entry application. The pre-populated functions could then be associated with specific artifacts within the entry application and reduce the overall workload of the user.

6 REFERENCES

1. Bohm, M., R. Stone and S. Szykman, 2005, "Enhancing Virtual Product Representations for Advanced Design Repository Systems," Journal of Computer and Information Science in Engineering, **5** (4): 360-372.
2. Svensson, D. and J. Malmqvist, 2001, "Integration of Requirement Management and Product Data Management Systems," Proceedings of DETC2001, Pittsburgh, PA, **DETC2001/CIE-21246**.
3. Regli, W. C. and D. M. Gaines, 1997, "A repository for design, process planning and assembly.," Computer-Aided Design, **29** (12): 895-905.
4. Summers, J., D. Maxwell, C. Camp and A. Butler, 2000, "Features as an Abstraction for designer convenience in the Design of Complex Products," Proceedings of DETC-2000, Baltimore, MD, **DETC2000/CIE-14642**.
5. Dixon, J., E. Libardi, S. Luby, M. Vaghul and M. Simmons, 1987, "Expert Systems for Mechanical Design: Examples of Symbolic Representations of Design Geometries," Engineering Computing, **2** (1): 1-10.
6. Sturges, R., K. O'Shaughnessy and M. Kilani, 1996, "Computational model for conceptual design based on extended function logic," Artificial Intelligence for Engineering Design, Analysis and Manufacturing, **10** (4): 255-274.
7. Bohm, M. R., R. B. Stone, T. W. Simpson and E. D. Steva, 2006, "Introduction of a Data Schema: The Inner Workings of a Design Repository," Design Engineering and Technical Conference 2006, Philadelphia, PA, **DETC2006/CIE-99518**.

8. Dalheimer, M. K., 2002, Programming with Qt, Second Edition, O'Reilly, London.

9. Kurtoglu, T., M. Campbell, C. Bryant, R. Stone and D. McAdams, 2005, "Deriving a Component Basis for Computational Functional Synthesis," International Conference on Engineering Design, ICED'05, Melbourne, Australia.

10. Greer, J. L., M. E. Stock, R. B. Stone and K. L. Wood, 2003, "Enumerating the Component Space: First Steps Toward a Design Naming Convention for Mechanical Parts," ASME Design Engineering Technical Conference, Design Theory and Methodology Conference, Chicago, IL, **DETC2003/DTM-48666**.

11. Hirtz, J., R. Stone, D. McAdams, S. Szykman and K. Wood, 2002, "A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts," Research in Engineering Design, **13** (2): 65-82.

12. Bohm, M. and R. Stone, 2004, "Representing Product Functionality to Support Reuse: Conceptual and Supporting Functions," Proceedings of DETC2004, Salt Lake City, UT, **DETC2004-57693**.

13. Nagel, R. L., M. R. Bohm and R. B. Stone, 2007, "A Representation of Carrier Flows for Functional Design.," Accepted to International Conference on Engineering Design, ICED'07, Paris.

14. Stone, R., I. Tumer and M. Stock, 2005, "Comparing Two Levels of Functional Detail for Mapping Historical Failures: You Are Only as Good as Your Knowledge Base," Research in Engineering Design, **In Press, DOI 10.1007/s00163-005-0005-z**.

15. Stone, R. B., I. Tumer and M. Van Wie, 2004, "The Function Failure Design Method," Journal of Mechanical Design, **127** (3): 397-407.

16. Grantham Lough, K., R. Stone and I. Tumer, 2005, "Function Based Risk Assessment: Mapping Function to Likelihood," Proceedings of DETC'05, Long Beach, CA.

17. Lock, D., 1993, *Handbook of Engineering Management*, Butterworth-Heinemann, Oxford.

3. USING A DESIGN REPOSITORY TO DRIVE CONCEPT GENERATION

Matt R. Bohm
Jayson P. Vucovich
and Robert B. Stone, Ph.D.
Design Engineering Laboratory
Department of Interdisciplinary Engineering
University of Missouri – Rolla
Rolla, Missouri 65409

ABSTRACT

This paper describes how a design repository can be used as a concept generation tool by drawing upon archived function-based design knowledge. Modern design methodologies include several types of activities to formally generate design concepts. Typical concept generation methods range from open-ended creative brainstorming activities to quantitative function-component analysis. A combination of two such methods—the Chi Matrix and Morphological Matrix techniques—is the basis for this work. Building on existing functionality of the design repository, desired product functions can be specified in a search of stored design knowledge, returning a Morphological Matrix of artifacts solving the specified functions. Such a search is termed a Morphological Search. The repository Morphological Search feature is evaluated against concepts generated in a previous original design project,.

Results of the Morphological Search return are then compared to ten of the original concept variants generated during the design project. This comparison shows that 89% of the specified subfunctions return results and that, on average, 77% of the components used in the hand-generated concepts can be derived by using the Morphological Search feature.

1 INTRODUCTION

As the product development space becomes more complex and competitive, it is essential that designers have a wide variety of tools to aid in the many aspects of product development. Many tools exist that all specialize in certain aspects of the product development space. For example, CAD tools allow designs to be visualized and moved to production, while Finite Element Analysis (FEA) packages allow for specific components to be structurally analyzed. While many design packages exist for the CAD and FEA space of product development, few software packages are geared toward the pre-form space of product development. One such tool aimed towards the pre-form phase of product development is a design repository which are used to archive, store, and retrieve existing design knowledge in a formalized method.

Over the span of several years of research and integrated design coursework, a web-based design repository has been implemented at the University of Missouri-Rolla. The design repository currently contains detailed design knowledge for over 100 consumer electromechanical products and provides an interface for user specified searches across all products. With this knowledge-base in place, tools are being built to utilize the repository infrastructure to support conceptual product development. By expanding the design repository's search capability we present a tool aimed specifically toward concept generation activities.

2 BACKGROUND

Modern product design techniques demand that the designer spend an increasing amount of time and effort to research possible design solutions, draw upon many disciplines and backgrounds, and often reach outside the designer's own domain of experience. Design repositories have the capacity to store and retrieve design knowledge such that the designer can have easy access to a wide array of design solutions beyond his or her own stored knowledge. Such repositories benefit from well defined taxonomies. In particular, a functional taxonomy known as the

Functional Basis [1] and a component naming infrastructure [2, 3] are utilized in the UMR design repository. The description language for artifacts in a repository increases the computability of design information and eliminate ambiguities between individuals sharing information. Using the Functional Basis to represent product functionality within the design repository allows product knowledge to be searched and categorized by their function. This abstraction allows the designer to focus on overall functionality and to develop more creative solutions for solving a design problem [4]. For this paper, we will be considering the design repository under development at the University of Missouri-Rolla Design Engineering Lab [5, 6, 7].

2.1 Concept Generation Techniques

Many researchers have sought to formalize the conceptual design phase. Antonsson and Cagan concisely define the notion of 'formal' as "...computable, structured, and rigorous, not ad hoc" [8]. Furthermore, by founding concept generation techniques on functionality, solution-independent design descriptions can be built [9]. Such methods generally rely on a form of functional decomposition of the overall problem to initiate the search for physical design solutions during conceptual design. Whether driven in this function-based manner or otherwise, much variability is exhibited in just how this search is carried out depending on the method chosen. This reflects the variety of perspectives that have been suggested for addressing the conceptual design problem.

2.1.1 C-Sketch/6-3-5 Method

The 6-3-5 method is a generic technique that supports innovative thinking [10]. In 6-3-5, members of an engineering design team (optimally 6-8 members) generate, interpret, and modify the individual ideas of other team members by first brainstorming and sketching individually on three ideas for various aspects of the product, then passing their ideas to the next team-member who adds additional ideas and sketches. C-Sketch is a variant of the 6-3-5 method wherein members produce only sketches and

refrain from communicating verbally when passing ideas to the next member. Passing only sketches allows other team members the opportunity to interpret the concepts in a different manner than the original author, thereby increasing design diversity.

2.1.2 Design by Analogy

In Design by Analogy, a functional model is created of the product being designed. Examining analogous products or components that perform the same function generates solutions to the present design problem. The designer then evaluates these similar components for appropriateness in solving the given design problem [4].

One Design By Analogy method widely recognized in the engineering design community is the Theory of Inventive Problem Solving, or simply TRIZ. TRIZ was developed by Altshuller during the 1940-50's period and was based on the examination of large numbers of existing patents [11]. The end result of this effort is an engineering design approach that identified a set of specific conflicts that occur in design along with a set of principles that can be applied to generate solutions that solve these conflicts.

2.1.3 Morphological Matrix Method

The morphological matrix introduced by Zwicky is a now a classic technique for use in conceptual design [12]. This method provides the design engineer with a simple, albeit manual, means for bookkeeping potential physical solutions and their corresponding functionality.

A morphological matrix is traditionally created by listing all of the sub-functions for a design and brainstorming solutions to each sub-function, listing the solutions as columns and the sub-functions as rows [10, 13, 14, 15, 16].

In a manual engineering design context, the morphological matrix is limited to the concepts generated by the engineer, although the morphological matrix is one technique that can be used in conjunction with overall design processes such as 6-3-5 or the reverse engineering and redesign method of [10].

2.1.4 Chi-Matrix Method

The chi-matrix method relies on a catalog of design information that stores components and the functions they perform [17]. When a designer desires to generate concepts for a given design problem, a filter matrix is used which contains only the functions needed for the given problem. This filter is multiplied into the aggregate function-component matrix to produce a matrix that contains only components that solve the needed functions. In this way a designer can generate possible solutions without having to search the entire store of knowledge manually.

3 DESIGN PROJECT

Before discussing the Repository Morph Search further, we introduce a design problem that uses traditional concept generation techniques from Section 2. The goal of this design problem was to transform an imprecise counting and packaging line at the Rolla Area Sheltered Workshop. The solutions generated for that design problem are used here to compare the results of manual concept generation techniques with the results of an automatically generated morphological matrix using a design repository. The device, prototyped at University of Missouri – Rolla (UMR), was the product of several modern design methodologies. Initial customer interviews were conducted, a customer needs questionnaire was developed, technical requirements were formed, and several types of concept generation and selection techniques were applied to this original design project.

3.1 Case Project Background

The Rolla Area Sheltered Workshop employs persons with mental and physical disabilities to package variety boxes of dog and cat food sample packets for a local pet food manufacturer. In the interest of increased productivity and a reduced incidence of repacking, a counting and packaging assistive device was sought. The design team began by observing the previous method of packaging used by the employees. At this point the Workshop did not have any specific solution sets in mind. An informal two-

way question & answer session took place between the design team and Workshop managers so that both groups had an understanding of the problem and what types of design solutions would be valid.

3.2 Functional Model

A functional model is a description of a product or process in terms of the elementary functions that are required to achieve its overall function or purpose. A graphical form of a functional model is represented by a collection of sub-functions connected by the flows on which they operate. This structure is an easy way for a designer to see what type of functions are performed without being distracted by any particular form the artifact may take. A functional model of the dog food packaging device is shown in Fig. 1.

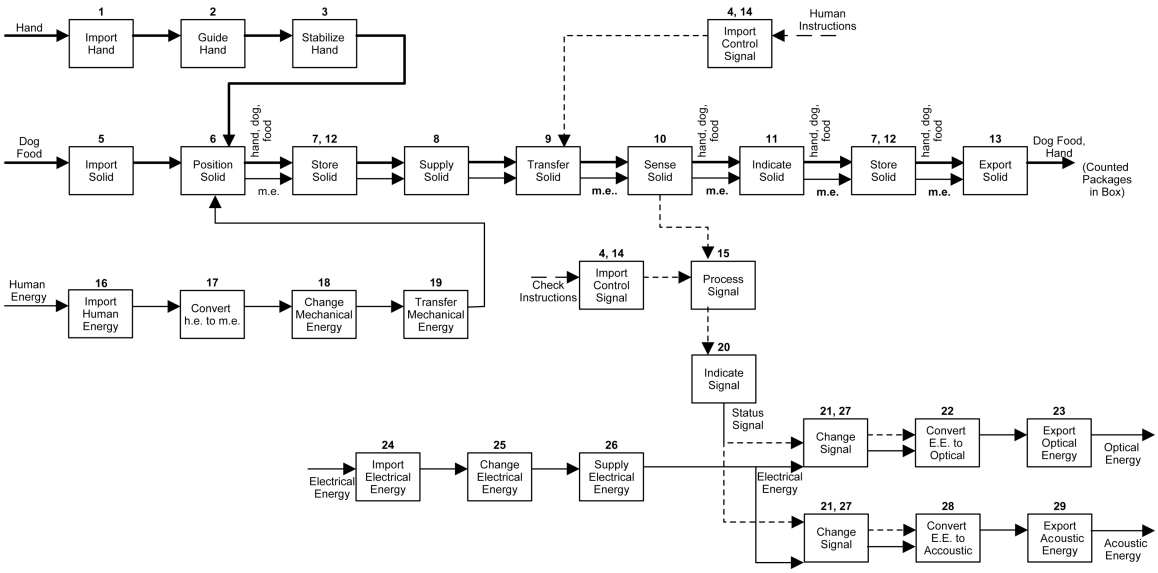


Figure 1. Functional Model

3.3 Concept Generation

The functional model is a useful tool during the concept generation phase of the project. Because all of the required functions are identified, the design team can focus on

developing solutions for given functions one at a time resulting in an entire concept. There are several formal methods for developing concepts which are designed to help stimulate a design team's creativity [18, 19]. In particular, the methods employed during this project were the C-sketch method, Design by Analogy, the Chi-Matrix approach and the Morphological Matrix approach. Using all of these methods allows for a broad spectrum of design concepts to be generated.

3.3.1 Concepts Generated by the C-sketch Method

By using the C-sketch method, the design team was able to generate five design concepts. Three of the concepts were based on mechanical and electrical systems to transport and count the dog food packets. The fourth concept contained no moving parts or electronics and was a simple plastic tray with color-coded slots. The fifth concept built on concept 4 by adding switches and buzzers to indicate when the slots were full. Figure 2 shows three such concepts developed using the C-Sketch method (C-sketch 1, C-sketch 3 and C-sketch 5, respectively).

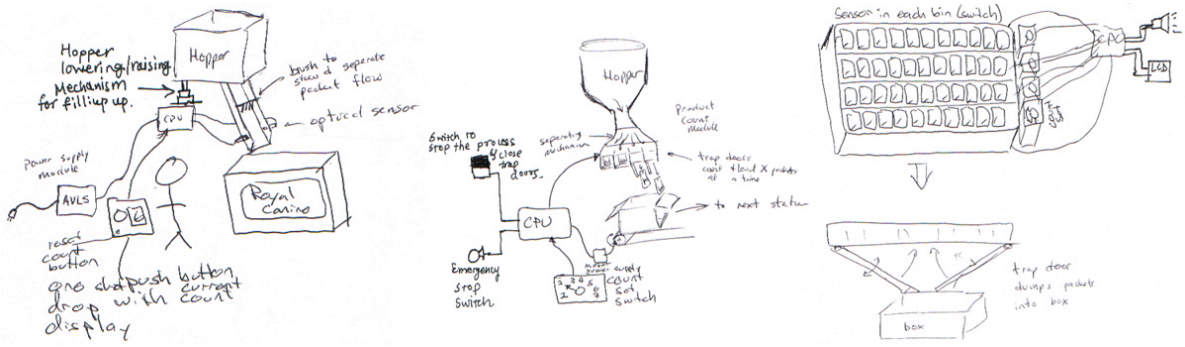


Figure 2. Concepts Generated by the C-Sketch Method

3.3.2 Concepts Generated with Design by Analogy

Four concepts were produced using the Design by Analogy method. The first three concepts were electro-mechanical devices using conveyors and sensors to count

and transport the dog food packets. The forth concept was a plastic tray variant with rotating handles to empty the counted dog food packets directly into the box.

3.3.3 Concepts Generated by the Chi-Matrix Method

Employing the Chi-Matrix approach generated five concepts. The first concept was based on a case with individual dog food packet receptacle slots. A sliding door was placed beneath the receptacles and was used to empty the slots once they are filled directly into the packing box via a chute. The remaining 4 concepts incorporated fairly simple electronics to act as counters while dog food packets were manually placed in the box. Figure 3 shows concepts Chi-Matrix 1, Chi-Matrix 2, and Chi-Matrix 5 as example solutions generated by the design team using this method. Although the method is similar to the Morphological Matrix Search method discussed in the remainder of this paper, the Chi-Matrix solutions here were produced by hand using a different set of data.

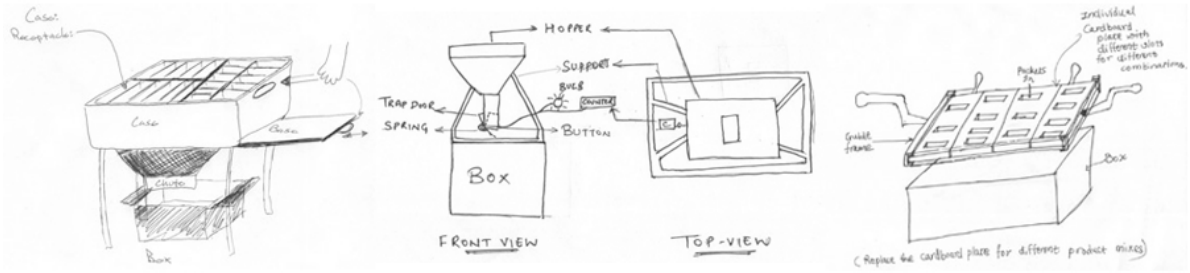


Figure 3. Concepts Generated by the Chi-Matrix Method

4 USING THE REPOSITORY MORPHOLOGICAL SEARCH FEATURE

As a test case, subfunctions identified by a customer needs based functional model from the bulk-packaging device design project (introduced in Section 3) are used in the Morphological Search feature. The returned search results are then compared to original bulk-packaging device concepts. Two sets of test data are presented in this work. The first set of test data was created using a repository containing only 29

products discussed as trial 1. The second set of test data, trial 2, comes from the same repository but containing 68 products. The premise of this comparison is that if the Morphological Search tool can generate concepts that match the results of the design team (produced by following the creativity-based concept generation techniques), then the Morphological Search tool offers an automated approach that leads to at least as creative results as a design team.

4.1 Searching the Repository

The original functional model (shown previously in Fig. 1) developed for the bulk-packaging device contains 29 subfunctions (26 of them unique), which are summarized in Table 1 and numbered based on their order of appearance in the functional model.

Upon logging into the design repository, located at http://function.basiceng.umr.edu/repository, the user is presented with an options menu. To perform a Morphological Search, the user navigates to the Search page and is presented with the option to perform either a "Standard Artifact Search" or a "Morphological Chart Search". Once "Morphological Chart Search" is selected, the user is then presented with the Morphological Search options shown in Fig. 4.

A list of available products is presented on the left hand side of the Morphological Search Input. The user can select any combination of the products listed depending on their desired search domain. With the search base selected, the user then selects the number of subfunctions they wish to enter through the "Subfunction:" pull-down menu. At this time, a maximum of 10 subfunctions can be entered for a single search. If more than 10 subfunctions exist, the user must perform multiple searches. Once the number of subfunctions is selected, the user must specify the number of columns they wish to appear in the search return. A maximum of 20 columns can be displayed although 10 columns typically capture most, if not all, of the possible returns.

The user can now begin to specify the subfunctions they wish to search for by using the pull-down menus. Subfunctions are entered as a tuple representing the input flow, subfunction and output flow. The first subfunction entered in Fig. 4 relates to "import human material" but is specified in the format (human material, import, human material). For most functions, the input and output flow are identical; however, the input and output flow for some functions (e.g. convert) are different. Currently the subfunction and flow inputs used in the Morphological Search are limited to the secondary term of the Functional Basis [1]. If a primary or tertiary form of the Functional Basis is desired as search input, the user should render the function or flow in the secondary level Functional Basis.

Table 1. Identified Subfunctions

Subfunction Number	Subfunction
1	import hand
2	guide hand
3	stabilize hand
4, 14	import control signal
5	import solid
6	position solid
7, 12	store solid
8	supply solid
9	transfer solid
10	sense solid
11	indicate solid
13	export solid
15	process signal
16	import he
17	covert he to me
18	change me
19	transfer me
20	indicate status signal
21, 27	change signal
22	convert ee to optical
23	export optical energy
24	import ee
25	change ee
26	supply ee
28	convert ee to acoustic
29	export acoustic

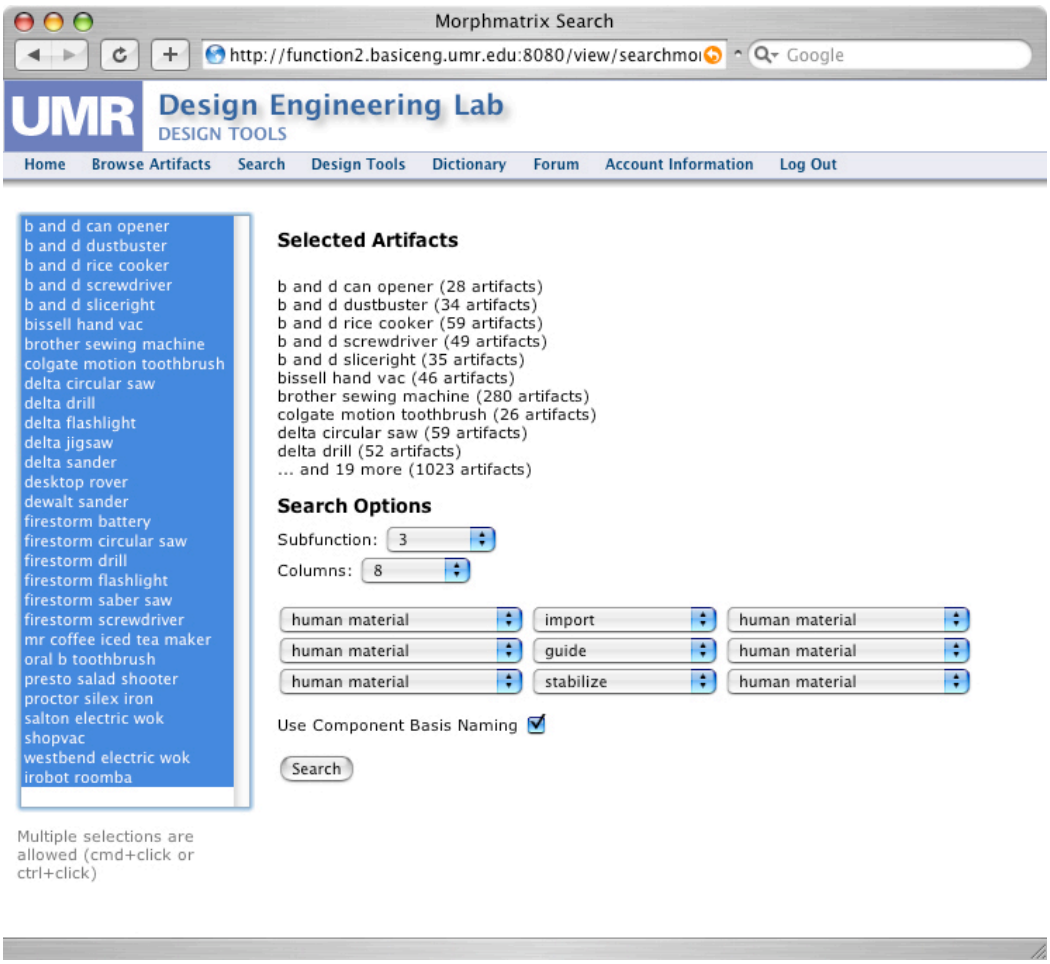


Figure 4. Morphological Search Input

With all of the desired subfunction tuples entered, the user can utilize the "Use Component Basis Naming" checkbox to choose how search results are returned. Checking the box categorizes returned artifacts into the component basis [2, 3]. Leaving the box unchecked will return results categorized by the name given to a specific artifact. For example, artifacts may be named "motor," "electric motor" and "dc motor," but they are all categorized by the component basis as "Electric Motor." Choosing to categorize search results by the component basis will group all instances of an electric motor as "Electric Motor." Without using the component basis categorization, the instances of "motor," "electric motor" and "dc motor" would be returned distinctly.

Upon submitting the search, a new browser window is opened containing the search results. These results for the three example subfunction tuples entered above in Fig. 4 are shown in Fig. 5. The left-most column of the results page displays the subfunction search criteria and subsequent columns (up to the amount specified) show the groupings of artifacts solving the given function. The results are sorted within each row by their rate of return. For example, a "Housing" of some sort is found to solve "Import Human Material" in 34.55% of the total number of solutions to "Import Human Material."

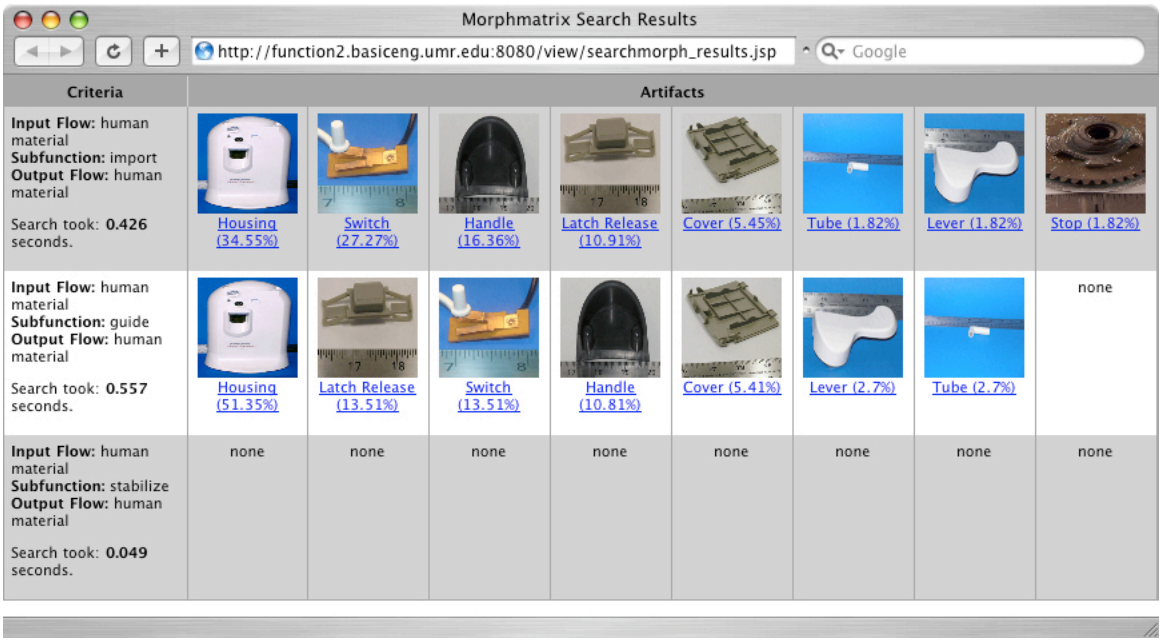


Figure 5. Morphological Search Results

For this particular search, results were returned for "import human material" and "guide human material" while no artifacts were found for the "stabilize human material" criteria. To view specific instances of a returned component grouping, the user can click on the link below the component image. Figure 6 shows all of the 19 artifacts

classified as a "Housing" for the "import human material" search criteria. Listed along side each artifact is the artifact's parent product. For example, the "Left Case Handle" artifact originated from the Black&Decker Dustbuster. If the user wishes to view more information about a specific artifact, they can do so by clicking the artifact name.

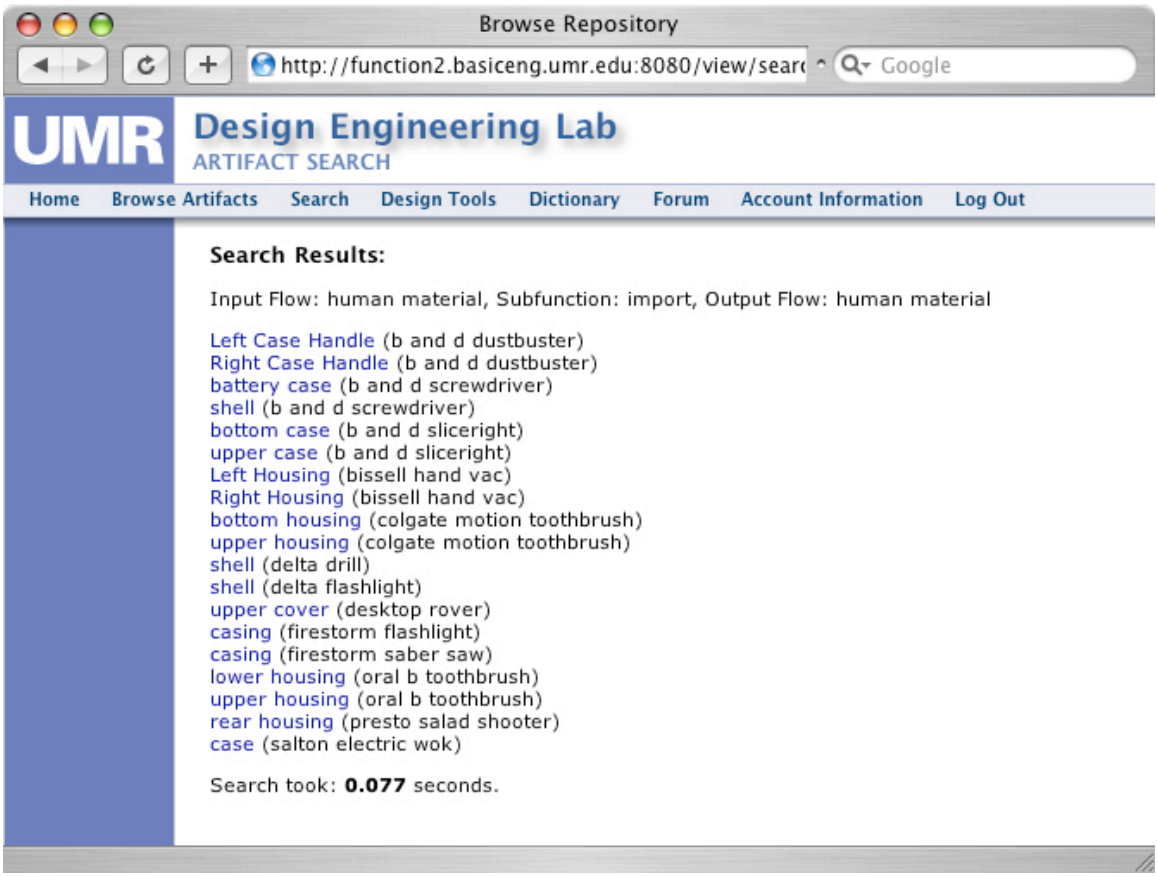


Figure 6. Detailed Component List for Housing

4.2 Distilling the Results

Morphological Searches were carried out for the remaining 26 subfunctions identified for the bulk-packaging device. Table 2 summarizes the Morphological Search results based only on function return. Out of the 29 subfunctions searched, trial 1 returned results for 21 of the functions while trial 2 returned results for 26 of the

functions. The function return results correspond to 72% and 89% for trials 1 and 2, respectively.

Table 2. Results Based on Function Return

	function	trial 1	trial 2
1	import human material	x	x
2	guide human material	x	x
3	stabilize human material		
4	import control signal	x	x
5	import solid material	x	x
6	position solid material	x	x
7	store solid material	x	x
8	supply solid material	x	x
9	transfer solid material	x	x
10	sense solid material		
11	indicate solid material		
12	store solid material	x	x
13	export solid material	x	x
14	import control signal	x	x
15	process control signal		x
16	import human energy	x	x
17	convert human energy to mechanical energy	x	x
18	change mechanical energy	x	x
19	transfer mechanical energy	x	x
20	indicate status signal		x
21	change status signal		x
22	convert electrical energy to optical energy	x	x
23	export optical energy	x	x
24	import electrical energy	x	x
25	change electrical energy	x	x
26	supply electrical energy	x	x
27	change status signal		x
28	convert electrical energy to acoustic energy		x
29	export acoustic energy	x	x
	totals:	21	26

Ten of the 31 concepts developed during the bulk-packaging device project were chosen to compare to the Morphological Search results. The concepts are named for the technique that was used for their generation. For example, “Chi-Matrix 1” corresponds to the first concept developed by using the Chi-Matrix approach. The concepts named “Chi-Matrix 1”, “Chi-Matrix 2”, “Chi-Matrix 4”, “Chi-Matrix 5”, and “C-Sketch 5” were identified by the original design team as their top-five concepts. The remaining concepts

were selected from the pool of 31 because they represented complete design solutions with definable functionality and were well-documented.

In order to compare the Morphological Search results to the concepts developed for the bulk-packaging device, concept sketches and notes were revisited. Since the subfunctions used for the Morphological Search input originated from the initial functional model of the bulk-packaging device, each concept was then related to the same set of subfunctions. There are some differences between the subfunctions identified in each of the concepts and those of the original functional model. The subfunction variation is due to the natural progression in the design process where customer needs are refined and the product direction is better identified. Table 3 shows a mapping of the originally identified subfunctions to each of the concepts used in this study.

Components that solve each subfunction found in the concepts are identified, completing the comparison to the Morphological Search results. Table 4 shows the identified subfunctions and components for the Chi-Matrix 1 concept comparison in trial 2. Note that the components listed in the columns represent only those components that

were identified as part of the Chi-Matrix 1 concept. Components that were identified to solve a specific function are denoted with a '1' while a shaded function-component combination shows that no results were returned for the combination by the Morphological Search. For the Chi-Matrix 1 concept, 15 components were identified but the Morphological Search only returned 12 of the same function mapped components resulting in an 80% return of components.

To quantify the amount of similarity between the concept's function-component matrix (**C**) and the function-component matrix (**R**) returned by the Morphological Search, a simple routine is devised. After each of the two matrices are converted to binary matrices, an overlap table is constructed by multiplying $Cij * Rij$ only if $Cij=1$. In this manner, a table is built containing null values if the product matrix does not contain

the given function-component pair, a one if both the product matrix and the repository matrix have the function-component pair, and a zero if the product contains the function-component pair but no match is found in the repository matrix. In cases where the returned Morphological Matrix **R** does not contain a component used in **C**, a "Not Found" column is used with a zero in **R** and a one in **C**. Using the totals gained from the overlap table, a simple ratio of total pair-matches (the ones) to total overlap (the zeros and ones) is calculated. This ratio represents the percentage of the product's functionality that is captured by the repository search.

Table 3. Identified Subfunctions of Concepts

	Dominate Functions Identified in Individual Concepts	Chi-Matrix 1	Chi-Matrix 2	Chi-Matrix 4	Chi-Matrix 5	C-Sketch 1	C-Sketch 2	C-Sketch 3	C-Sketch 4	C-Sketch 5	DBA 1
1	import human material	**1**	0	0	**1**	0	**1**	0	**1**	0	**1**
2	guide human material	**1**	0	0	**1**	0	**1**	0	0	0	**1**
3	stabilize human material	**1**	0	0	**1**	0	**1**	0	0	0	**1**
4	import control signal	0	0	0	0	0	0	**1**	**1**	**1**	0
5	import solid material	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
6	position solid material	**1**	**1**	**1**	**1**	**1**	0	**1**	**1**	**1**	0
7	store solid material	**1**	**1**	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**
8	supply solid material	**1**	**1**	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**
9	transfer solid material	**1**	**1**	0	**1**	**1**	**1**	**1**	0	0	0
10	sense solid material	0	**1**	**1**	0	**1**	**1**	**1**	0	**1**	0
11	indicate solid material	**1**	**1**	**1**	0	**1**	**1**	0	0	**1**	**1**
12	store solid material	**1**	**1**	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**
13	export solid material	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	**1**
14	import control signal	0	0	0	0	0	0	**1**	**1**	**1**	0
15	process control signal	0	**1**	**1**	0	**1**	**1**	**1**	0	**1**	0
16	import human energy	**1**	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**
17	convert human energy to mechanical energy	**1**	0	0	**1**	0	0	0	0	0	**1**
18	change mechanical energy	**1**	0	0	**1**	0	0	**1**	0	0	**1**
19	transfer mechanical energy	**1**	0	0	**1**	0	0	0	0	0	0
20	indicate status signal	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
21	change status signal	0	**1**	**1**	0	**1**	**1**	**1**	0	**1**	0
22	convert electrical energy to optical energy	0	**1**	**1**	0	0	0	**1**	0	0	0
23	export optical energy	0	**1**	**1**	0	0	0	0	0	0	0
24	import electrical energy	0	**1**	**1**	0	**1**	**1**	**1**	0	**1**	**1**
25	change electrical energy	0	**1**	**1**	0	**1**	**1**	**1**	0	**1**	**1**
26	supply electrical energy	0	**1**	**1**	0	**1**	**1**	**1**	0	**1**	**1**
27	change status signal	0	**1**	**1**	0	**1**	**1**	**1**	0	**1**	0
28	convert electrical energy to acoustic energy	0	0	0	0	0	0	0	0	**1**	0
29	export acoustic energy	0	0	0	0	0	0	0	0	**1**	0

Table 4. Identified Components and Results (trial 2) for Chi-Matrix 1

	Chi Matrix 1 Results for Trial 2	Cover	Guide	Handle	Hinge	Housing	Reservoir	Support
1	import human material	0	0	**1**	0	0	0	0
2	guide human material	0	0	**1**	0	0	0	0
3	stabilize human material	0	0	**1**	0	0	0	0
4	import control signal	0	0	0	0	0	0	0
5	import solid material	0	**1**	0	0	0	0	0
6	position solid material	0	0	0	**1**	0	0	0
7	store solid material	0	0	0	0	**1**	0	0
8	supply solid material	0	**1**	0	0	0	0	0
9	transfer solid material	**1**	0	0	0	0	0	0
10	sense solid material	0	0	0	0	0	0	0
11	indicate solid material	0	0	0	**1**	0	0	0
12	store solid material	0	0	0	0	0	**1**	0
13	export solid material	0	**1**	0	0	0	0	0
14	import control signal	0	0	0	0	0	0	0
15	process control signal	0	0	0	0	0	0	0
16	import human energy	0	0	**1**	0	0	0	0
17	convert human energy to mechanical energy	**1**	0	0	0	0	0	0
18	change mechanical energy	0	0	**1**	0	0	0	0
19	transfer mechanical energy	0	0	0	0	0	0	**1**
20	indicate status signal	0	0	0	0	0	0	0
21	change status signal	0	0	0	0	0	0	0
22	convert electrical energy to optical energy	0	0	0	0	0	0	0
23	export optical energy	0	0	0	0	0	0	0
24	import electrical energy	0	0	0	0	0	0	0
25	change electrical energy	0	0	0	0	0	0	0
26	supply electrical energy	0	0	0	0	0	0	0
27	change status signal	0	0	0	0	0	0	0
28	convert electrical energy to acoustic energy	0	0	0	0	0	0	0
29	export acoustic energy	0	0	0	0	0	0	0

Table 5 shows the comparison between specific concepts and the Morphological Search results for trials 1 and 2. For the concept Chi-Matrix 1, 71.43% of the components used in the concept were returned by the Morphological Search in trial 1 but increases to 80% in trial 2. This means that 80% of the concept could have been derived by using the Morphological Search feature of the repository. Analysis of all of the concepts for trial 2 indicate that an average of 77.07% of the ten manually derived concepts could have been automatically generated by the repository's design tools system. A mature repository could conceivably generate 100% of the manually generated concepts.

Table 5. Component Similarity Results

concept	trial 1	trial 2
Chi-Matrix 1	71.43%	80.00%
Chi-Matrix 2	62.50%	75.00%
Chi-Matrix 4	61.54%	69.40%
Chi-Matrix 5	71.43%	85.70%
C-Sketch 1	53.33%	81.25%
C-Sketch 2	53.33%	81.25%
C-Sketch 3	58.82%	73.60%
C-Sketch 4	66.67%	77.80%
C-Sketch 5	50.00%	66.67%
DBA-1	66.67%	80.00%
Average:	*61.57%*	*77.07%*

5 CONCLUSIONS

The Morphological Search tool offers designers an additional approach for generating concept variants and presents historically recorded subfunction solutions in the familiar morphological matrix format. Given that the empirical case study finds that 77% of the concepts reviewed can be derived using the Morphological Search tool we conclude that the method shows promise as an automated concept generation tool. The use of the Component Basis in this evaluation ensures the reliability of these results by standardizing the comparison basis between the two projects.

Additionally, the high level of commonality between these automatically generated concepts and handgenerated concepts contributes to the notion of utility for such a design tool. The 89% average return of functionality for this case study demonstrates that a relatively small number of products (68 in this case) can constitute a useful and usable design repository.

Comparing the Morphological Search tool to previously generated concepts is a novel, systematic test to demonstrate the suitability of the returned results. Overall, our results add supporting evidence to ongoing work that attempts to show the utility of computational-based conceptual design methods. Specifically, the results show that the Morphological Search feature can be an effective tool capable of replicating a substantial amount of solutions automatically that would otherwise be generated by hand in state-

of-the-art conceptual design techniques. As with any concept generation technique, actual usefulness of the Morphological Search tool depends greatly upon the designer's ability to gain insight from the tool.

As the knowledge base grows, the potential number of concepts suggested by the Morphological Search tool also grows. This effect is directly related to the number of distinct subfunctions contained in the repository. Until a distinct subfunction bound is reached, the number of concepts suggested by the search tool will increase. From one perspective, this is a desirable result. At the early stages of design, it is beneficial to generate as many concepts as possible. From a different perspective, evaluating all of the concepts becomes a burdensome and time-consuming task. Compatibility reasoning methods, domain similarity or other computational techniques aimed at reducing the set of suggested concepts to some "best" subset for detailed review by the designer remains as future work.

ACKNOWLEDGEMENTS

This work is supported by the National Science Foundation under grant and IIS-0307419. Any opinions or findings of this work are the responsibility of the authors and do not necessarily reflect the views of the sponsors or collaborators.

REFERENCES

[1] 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.

[2] Greer, J. L., Stock, M. E., Stone, R. B. and Wood, K. L., 2003, "Enumerating the Component Space: First Steps Toward a Design Naming Convention for Mechanical Parts," DETC03/DTM-48666, Proceedings of DETC2003, Chicago, IL.

[3] Kurtoglu, T., Campbell, M., Bryant, C., Stone, R. and McAdams, D., 2005, "Deriving a Component Basis for Computational Functional Synthesis," International Conference on Engineering Design, ICED'05, Melbourne, Australia.

[4] McAdams, D. and Wood, K., 2000, "Quantitative Measures for Design By Analogy," DETC2000/DTM-14562, Proceedings of DETC2000, Balitmore, MD.

[5] Bohm, M. and Stone, R., 2004, "Representing Product Functionality to Support Reuse: Conceptual and Supporting Functions," DETC2004-57693, Proceedings of DETC2004, Salt Lake City, UT.

[6] Bohm, M. and Stone, R., 2004, "Product Design Support: Exploring a Design Repository System," IMECE2004-61746, ASME International Mechanical Engineering Congress, Anaheim, CA.

[7] Bohm, M., Stone, R. and Szykman, S., 2003, "Enhancing Virtual Product Representations for Advanced Design Repository Systems," DETC2003/CIE-48239, Proceedings of DETC2003, ASME, Chicago, IL.

[8] Antonsson, E. and Cagain, J., 2001, Formal Engineering Design Synthesis, Cambridge University Press, New York, NY.

[9] Pahl, G. and Wallace, K., 2002, Using the Concept of Functions to Help Synthesize Solutions, Springer, London.

[10] Otto, K. and Wood, K., 2001, Product Design: Techniques in Reverse Engineering, Systematic Design, and New Product Development, Prentice-Hall, New York.

[11] Altshuller, G., 1984, Creativity As An Exact Science, Gorden and Breach, Luxembourg.

[12] Zwicky, F., 1969, Discovery, Invention, Research - Through the Morphological Approach, The Macmillian Company, Toronto.

[13] Pahl, G. and Beitz, W., 1996, Engineering Design: A Systematic Approach, Springer-Verlag, London, UK.

[14] Ullman, D. G., 1997, The Mechanical Design Process, McGraw-Hill, New York, NY.

[15] Ulrich, K. T. and Eppinger, S. D., 1995, Product Design and Development, McGraw-Hill, New York, NY.

[16] Hubka, V. and W. Ernst Eder, Theory of Technical Systems. 1984, Berlin: Springer-Verlag.

[17] Strawbridge, Z., McAdams, D. A. and Stone, R. B., 2002, "A Computational Approach to Conceptual Design," DETC02/DTM-34001, Proceedings of DETC2002, ASME, Montreal, Canada.

[18] Shah. J. (1998). "Experimental Investigation of Progressive Idea Generation Techniques in Engineering Design, " Proceedings of the 1998 ASME Design Theory and Methodology Conference, DETC98/DTM-5676, Atlanta, GA.

[19] Shah, J. J., Vargas-Hernandez, N., Summers, J. D. and Kulkarni, S., 2001, "Collaborative Sketching (C-Sketch) - An Idea Generation Technique for Engineering Design," Journal of Creative Behavior, 35(3), pp. 168-198.

4. A NATURAL LANGUAGE TO COMPONENT TERM METHODOLOGY: TOWARDS A FORM BASED CONCEPT GENERATION TOOL

Matt R. Bohm
mbohm@mst.edu
Design Engineering Lab
Department of Interdisciplinary Engineering
Missouri University of Science and Technology
Rolla, MO, 65409 USA

Robert B. Stone, Ph.D.
rstone@mst.edu
Design Engineering Lab
Department of Interdisciplinary Engineering
Missouri University of Science and Technology
Rolla, MO, 65409 USA

ABSTRACT

This paper reports on research leading to a natural language to component naming method that underpins an emerging form-initiated concept generation tool. The purpose of identifying standard component terms from natural language phrasing is to support computational parsing of an initial set of physical artifacts that solve a design problem as suggested in natural language by a novice designer. Parsing the natural language transfers the burden of design abstraction to the computer and more seamlessly integrates with existing concept generation algorithms. By leveraging an existing design repository data set and a hierarchical component naming taxonomy a detailed algorithm for natural language to component synonym identification is presented.

1 INTRODUCTION

From an engineering education standpoint, design is the perhaps one of the toughest topics to teach and, often, the most feared course assignment in a given engineering department. That is likely due to the emphasis of modern design techniques on abstracting the problem and identifying this fuzzy, hard to grasp (and explain) concept of functionality [1-4]. Yet, this is the area of engineering where innovation takes root and where students need the most nurturing.

At both the student and professional level, the major obstacle that designers face is the leap it takes to abstract a design problem to its constituent functionality – the essence, according to the above methodologies, to synthesizing the product that will meet customer demands. The natural language processing research presented underpins an alternative approach that is, based on over a decade of observation, more natural for engineering designers. The approach, which we will call Form Follows Form (FFF), automates concept generation by starting with suggested components that the designer believes may solve the design problem and extracts the underlying functionality of those components to create a set of more thorough and complete concept variants through existing concept generation algorithms [5-8].

1.1 Motivational Case

With the ability to translate a designer's natural language into a standardized, parse-able set of terms, designers would be allowed to build up chains of components they envision being in a new product. This is a task that initially appears to be simple, however, there are several ways one could describe information about components – from a topological adjacency matrix from such as a design structure matrix to a simple listing of components. Since ease of use and accessibility are key to this research, asking a user to first generate an adjacency matrix would be a cumbersome task. Alternatively, a simple list of components may not effectively capture the intent of the user (that is component connection and ordering). From a computational standpoint, information regarding components needs to, at a minimum, infer how the user intends those components to be connected to one another. For example, a user lists out wire, shaft and motor as components in a concept. A basic search of an appropriate design knowledge base would show that those components have not been observed to connect in that particular order. It is therefore necessary to build a framework that allows for an easy and logical manner to gain information about components in a particular concept. With

a semi-logical ordering of input components, algorithms will be better positioned to statistically determine the intended component order.

Figure 1 shows a potential interface to capture a perceived concept for the form-initiated concept generation approach. The example shown in Figure 1 contains the basic components of an iced tea maker. Users would be asked to enter components as a series of discrete chains. For example one chain of components may include a cord, switch, and a heating element while another may consist of a tank, tube and a condenser. Once chains are entered a user would be allowed edit, remove or reorder specific chains or components.

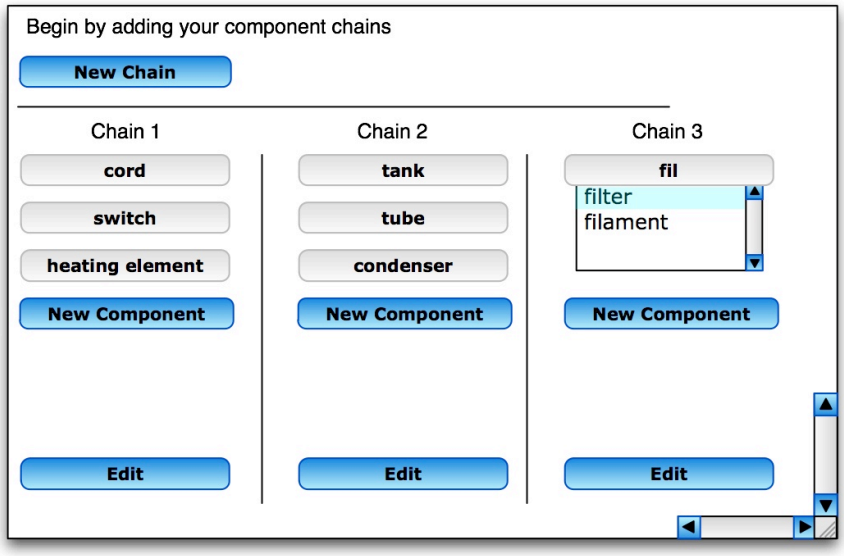


Figure 1. Mockup Envisioned Component Entry System

The envisioned components and their connection information can then be passed to an algorithm to develop a functional model of the product. Overall product functionality will be disguised from the user. The functional model will then be passed to existing computational concept generators. Computerized concept generation techniques, spanning the broad AI topics of knowledge representation and reasoning,

promise engineers a faster realization of potential design solutions based upon previously known products and implementations. FFF will be compatible with two existing concept generators. One of these methods utilizes relational matrices [9, 10] derived from the design repository while the other method relies on graph grammar rules [11, 12]

1.2 Relationship to Natural Language Interpretation in AI

The thrust of natural language interpretation in the artificial intelligence (AI) field is to provide a mechanism for machines to understand 'human speak' [13]. In the design context, it allows designers to specify components within a concept and do so using natural language. The current design repository makes use of a component naming taxonomy to classify artifacts with a general, standardized name. Artifacts within the repository can be tagged with a specific name such as "small dc motor" but are also tagged with the component naming term of "electric motor." This convention allows for artifact data to be clustered and analyzed but may also hinder how designers and engineers describe and search for a given component. If a designer were to search for a "tank" as a component naming term, using current implementation of the repository, no existing artifact would be found. This is because the word "tank" does not exist within the realm of the component naming terms. The term "tank," however, is a synonym of the component naming term "reservoir." In order to allow for designers to specify a concept using natural language it is necessary to attach additional synonym terms to the existing component naming terms.

The scope of this paper is to present our method to translate natural language component terms into standardized component terms as well as an initial set of natural language component synonyms. Both of these contributions are necessary to realize the overall research goal of FFF.

2 BACKGROUND

Three areas of prior work are necessary to support the natural language interpretation research of this work: design repositories, component naming terms and natural language interpretation as applied to engineering. Each topic is briefly reviewed next.

2.1 Design Repository

The objective of a Design Repository is to allow designers to store and retrieve design knowledge at various levels of abstraction, from form (components, sub-assemblies and assemblies) to architecture description to function. Currently the Design Engineering Lab's Design Repository contains design information for over 125 consumer based electro-mechanical products. Design information captured by the repository can be divided into seven main categories including: artifact-, function-, failure-, physical-, performance-, sensory- and media-related information types. The different levels of abstraction and types of design information provide innovative ways to approach design. With a well populated repository, emerging concept generator algorithms take, as input, basic product functionality or component information and instantaneously develop, filter and rank concepts to use as baselines for further product development. While the possibilities design repositories offer are diverse and helpful to designers, the implementation of such repositories are crucial to their overall success and usefulness.

Realizing the potential impact of an operational Design Repository, researchers at Missouri S&T, The University of Texas at Austin and the National Institute of Standards and Technology (NIST) began gathering artifact information in 1999 [14-16]. Since that time, the process in which artifact data is gathered and recorded has changed significantly. Initially, artifact design information was recorded in spreadsheets and mainly took the form of Bills of Materials (BOM), Function Component Matrices (FCM), and Design Structure Matrices (DSM). While this type of information was useful, it was also limited in scope and the required matrix multiplications were quite cumbersome. A

prior Design Repository initiative by NIST helped to guide the Design Repository project at Missouri S&T to a more mature state. To enhance data integrity, design information was migrated from spreadsheet form to a relational database. A web-based repository navigator including search and design tool generation features was created along with a repository entry application.

More recently, Missouri S&T has further partnered with UT-Austin [9, 10], Penn State [17], Virginia Tech, Bucknell [18], University of Buffalo and Texas A&M to expand the types of design information and breadth of design tool features within the repository. The Design Repository serves as a hub for designers for information exchange and design generation tools and is heavily utilized in the current VOICED project. Information entry and retrieval occurs within a standalone application [19] (available at http://designengineeringlab.org/repositoryEntry/) while information retrieval occurs over the Internet through the Design Repository's web portal (http://repository.designengineeringlab.org/). The infrastructure supporting these two applications is the Design Repository database and schema [17]. The database schema establishes what types of design information can be stored, the relationship of those elements and the extensibility of including new and additional types of design information.

2.2 Component Naming Taxonomy

The component naming taxonomy is a hierarchal naming system for engineering components [6, 7]. The taxonomy is a functional approach to component identification contains three levels of identification; 1.) the Primary Component Classification, 2.) the Secondary Component Classification, and 3.) the Component Term level. There are 8 different primary component classifications used to describe the generic function of a component. The secondary level adds specificity to the primary component level. For example, the primary level term channelers describes 3 secondary level terms: importers/exporters, transferors, and guiders.

Each secondary level term is then further decomposed to the component term level. Table 1 shows a portion of the component naming taxonomy for the component level terms of material suppliers (reservoir, container, bladder, and pressure vessel) and guiders (hinge, tube, diode, bearing, link, sled). Component terms are then associated with thesaurus derived synonyms and a detailed definition.

Table 1. Component Naming Taxonomy

Comp. Term	Synonyms	Definition
Reservoir	cup, vessel, bucket, bottle	A device in the form of an open tank used to accumulate and dispense a material.
Container	box, receptacle, holder	A device in the form of a closed canister used to accumulate and dispense a material.
Bladder	balloon	A device in the form of a hollow, expandable sac or membrane with a narrow opening used to accumulate and dispense a material.
Pressure Vessel	air tank, gas tank	A device in the form of a sealed tank used to accumulate and dispense a pressurized fluid material.
Hinge	pivot, axis, pin, hold down, jam, post, peg, dowel	A device that allows rigidly connected materials to rotate relative to each other about an axis, such as the revolution of a lid, valve, gate or door, etc.
Tube	pipe, cylinder, conduit, channel, duct, nipple, sleeve	A device in the form of a hollow body, usually cylindrical and long in proportion to its diameter, used to direct fluid material along a path.
Diode		A semiconductor device which allows current to flow in only one direction.
Bearing	journal bearing, thrust bearing	A device in the form of a ball or arrangement of balls that is placed between moving parts to allow them to move easily relative to each other along a path.
Link	connection, pawl, rod, strut, brace, cross piece, girder	A device connecting two or more components that transmits motive power from one part to another along a specific path.
Sled	shoe, runner, skid	A device either under or within a machine used to facilitate the sliding of components relative to each another along a path.

2.3 Natural Language in Engineering

The thrust of natural language interpretation is to remove formality as a requirement to computational activities and to stray away from specific terms and taxonomies. Recently there has been a great deal of work developing natural language terms to aid in biological inspired design [20-22]. Biological inspired design strives to bring elements and attributes that occur in nature to man-made products and processes. Unfortunately most engineers and designers know the language of engineering, not the language of biology. Natural language processing in this regard attempts to link what would be considered an engineering vocabulary to a biology/science vocabulary.

In order to develop a natural language translation for biological inspired design Chiu and Shu make use of keyword searches [20, 21]. Lexical references are then established by performing keyword searches on existing biological texts and articles. Without the wealth of existing published material it would be nearly impossible to develop such relationships. Chakrabarti et al. detail a method for developing analogies to link the natural and artificial world [22]. All of these works aim to better formalize biomimetic design, however, perhaps more fundamental is the goal making existing knowledge more accessible a synonym list or domain to domain thesaurus.

3 RESEARCH APPROACH: FORMULATING NATURAL LANGUAGE SYNONYMS

In this section, we examine repository data from a functional and artifact naming standpoint for two classes of the component naming taxonomy to guide the natural language interpretation activity. The section begins with a brief explanation of how the design repository stores artifact name, component naming term, and functional information. Next, Sections 3.2 and 3.3 examine functionality and naming statistics for both material suppliers and guiders.

3.1 Repository Conventions

The design repository contains over 5500 unique artifacts with over 99% of those artifacts also having a component naming term. Component naming information is

recorded using two separate database tables within the design repository. The artifact table allows for a common name to be associated with a particular artifact as well as point to a component naming term in the component_basis_type table. Designers who enter information in the design repository are allowed to specify an artifact's common name. There are no restrictions on an artifact's common name as long as it appropriately describe the component. Often times an artifact's common name will take the form "small dc motor" or "upper half case." Along with the common name designers are also allowed to specify a component term of the component naming taxonomy. For example an artifact with a common name of "small dc motor" should have a component name of "motor."

Functionality is recorded in a similar fashion and makes use of the functional basis for the allowed function and flow terms. Each artifact within the repository can have as few or as many associated functions and flows. A function-flow pair must be linked to an artifact and cannot be recorded independently.

3.2 A Functional Perspective

Next we examine how component naming terms are associated with functionality. The goal of this section is to determine the level of similarity or dissimilarity of component classes with regard to function. A high degree of function similarity within a component class would suggest component synonyms could be associated with the secondary level of the component naming terms. Significant dissimilarity, however, would suggest that component synonyms should be associated with the component level of the component naming taxonomy.

3.2.1 A Functional Look at Material Suppliers

Material suppliers include the component naming terms of bladder, container, pressure vessel and reservoir. At the time of analysis there were no artifacts in the repository labeled as a bladder, thus the term is removed from analysis. For the remaining components the top 6 functions for each are listed along with an incremental

percentage, shown in Table 2. The incremental percentage is a running percent total of overall function representation. For example, the function store represents 31.82% of all of the functions associated with the component naming term container. Likewise the functions store and import combined represent 48.86% of all functions associated with the component naming term container.

Table 2. Functions Associated with Material Suppliers

container		pressure vessel		reservoir	
function	Δ%	function	Δ%	function	Δ%
store	*31.82%*	supply	34.29%	*export*	*23.46%*
import	*48.86%*	*export*	*51.43%*	*import*	*41.15%*
export	*63.64%*	*store*	*68.57%*	*store*	*54.73%*
separate	71.59%	*import*	*82.86%*	stop	67.49%
mix	78.41%	couple	85.71%	position	72.43%
guide	82.95%	distribute	88.57%	supply	77.37%

The top 6 functions are shown for the container, pressure vessel and the reservoir. While performing the analysis we observe that in general 70% of all functionality is captured within the first 30% of the population of unique function terms. This phenomenon, known as Pareto Optimum, is better shown by Table 3 which contains a full listing of functions associated with a reservoir [23, 24]. The first column is the function term, the second column is the number of times that function is performed by a reservoir, the third column is a running percent of unique function terms (i.e., there are 15 unique terms each representing 6.67% of the population), the fourth column is the overall percentage a function exists when compared to the number of total functions, and the fifth column is a running summation of the fourth column. The function position is realized within approximately the first third of the population and at that point over 70% of all functions have been realized.

Table 3. Full Function Listing for Reservoir

reservoir				
function	# of observ.	% of population	% represent.	% summation
export	57	6.67%	23.46%	23.46%
import	43	13.33%	17.70%	41.15%
store	33	20.00%	13.58%	54.73%
stop	31	26.67%	12.76%	67.49%
position	12	33.33%	4.94%	72.43%
supply	12	40.00%	4.94%	77.37%
distribute	10	46.67%	4.12%	81.48%
secure	9	53.33%	3.70%	85.19%
couple	8	60.00%	3.29%	88.48%
mix	7	66.67%	2.88%	91.36%
transfer	7	73.33%	2.88%	94.24%
convert	6	80.00%	2.47%	96.71%
guide	5	86.67%	2.06%	98.77%
separate	2	93.33%	0.82%	99.59%
change	1	100.00%	0.41%	100.00%

Table 2 shows significant function overlap for material suppliers. The functions import, export and store are observed to occur for each component naming term. The function supply is also seen by 2 of the 3 components. This data suggests that material suppliers are overall functionally similar.

3.2.2 A Functional Look at Guiders

As with material suppliers it was noticed that in general 70% of all instances of function are realized within the first 30% of the population. Table 4 shows the full listing of functions associated with the component naming term tube in the same fashion as Table 3. There are 18 different functions solved by a tube with over 80% of all functions realized within the first third of the population. These results are a bit higher than most function realization relationships but are still considered to be in line.

Table 4. Full Function Listing for Tube

tube				
function	# of observ.	% of population	% represent.	% summation
import	76	5.56%	22.35%	22.35%
transfer	66	11.11%	19.41%	41.76%
export	64	16.67%	18.82%	60.59%
guide	38	22.22%	11.18%	71.76%
secure	19	27.78%	5.59%	77.35%
position	17	33.33%	5.00%	82.35%
couple	14	38.89%	4.12%	86.47%
stop	11	44.44%	3.24%	89.71%
distribute	9	50.00%	2.65%	92.35%
store	8	55.56%	2.35%	94.71%
convert	6	61.11%	1.76%	96.47%
change	4	66.67%	1.18%	97.65%
mix	2	72.22%	0.59%	98.24%
regulate	2	77.78%	0.59%	98.82%
indicate	1	83.33%	0.29%	99.12%
separate	1	88.89%	0.29%	99.41%
stabilize	1	94.44%	0.29%	99.71%
supply	1	100.00%	0.29%	100.00%

Table 5 summarizes all 6 guider component naming terms. Unlike material suppliers there is only a single function that appears for each of the component naming terms. The function guide appears at varying frequencies for each term, but still realized within the first 70% of all function instances. With the exception of diode, the remaining guiders are mostly similar duplicating the functions transfer, import, export, and guide. Conceptually this data is self supporting, you would not expect a diode to be similar to a hinge.

3.3 A Component Perspective

The method for component naming analysis is slightly more complex than the functional analysis. Complexity is introduced because users are allowed to name individual components while also assigning a component naming term. Common names for individual components may take the form of a single word such as “cup” or several words such as “lower left drip cup.” From a computational standpoint it is very

easy to aggregate single words of the same tense. It is much more difficult to automatically parse the phrase "lower left drip cup" into a single component name.

Table 5. Functions Associated with Guiders

bearing		diode		hinge	
function	Δ%	function	Δ%	function	Δ%
import	17.57%	change	33.33%	transfer	35.48%
export	33.78%	actuate	44.44%	*guide*	*51.61%*
guide	*45.95%*	convert	55.56%	export	64.52%
transfer	55.41%	*guide*	*66.67%*	import	77.42%
couple	63.51%	regulate	77.78%	convert	83.87%
secure	71.62%	sense	88.89%	couple	90.32%
link		sled		tube	
function	Δ%	function	Δ%	function	Δ%
transfer	58.12%	secure	40.63%	import	22.35%
guide	*66.67%*	transfer	56.25%	transfer	41.76%
import	75.21%	*guide*	*68.75%*	export	60.59%
export	80.34%	export	78.13%	*guide*	*71.76%*
convert	84.62%	import	87.50%	secure	77.35%
position	88.89%	convert	93.75%	position	82.35%

In order to analyze common component names with respect to their naming taxonomy a script was developed to first parse through the given common names. The script begins by creating a new database table to store alternate common names for a particular artifact. Any existing given component name that is a single word is then copied to the corresponding spot in the alternative name table. The script then prompts a user to enter new alternate names for the remaining common names. For example, a user would enter an alternate name of "cup" for the common name "lower left drip cup." The script then places "cup" as an alternate name for all artifacts with a common name of "lower left drip cup." Replacing all common name matches with the same alternate name allows for consistency in how data is interpreted. Approximately half of

the existing common artifact names were replaced with a new alternate name during this process.

A query was then structured to relate an artifact's component naming term to its new alternative name. One would expect for natural language terms to be associated with the secondary class of component terms if there is a high degree of overlap for component names. If there is a low or more singular relationship of common names and component naming terms it is proposed that the natural language synonyms be associated at the term level of the component naming taxonomy.

3.3.1 A Component Look at Material Suppliers

For analysis of material suppliers, only the component naming terms reservoir and container are included. The term bladder is not observed within the repository and the term pressure vessel is only seen 3 times, each time with a common name of pressure vessel. Because of the non-existent and limited data, Table 6 only examines common names associated with reservoir and container. Both terms are associated with 17 unique common names. Shown in italics are common names that are repeated between both component naming terms.

Again, the number next to a common name represents the number of times that common name has been associated with the corresponding component naming term. Since each term has 17 unique components their percent of the unique population column is the same, shown in the first column. The third column for each term is the percent representation of a common name seen across the entire data set and the fourth column is a running summation of the third column. Common names for reservoir again closely match the 70/30 Pareto optimality seen throughout the repository. Optimality (70/30) is not however recognized for container, but is on the lower bound of data seen across the repository.

In total, 76.9% of all instances of a container have also been denoted as a reservoir and 70% of all occurrences of a reservoir have also been labeled as a container. The high overlap suggests that users see the two component naming terms as synonyms. Recalling component naming term definitions detailed in Section 2.2, the main distinguishing factor between the two terms is that a container is closed while a reservoir is open. Perhaps a confusing point for users is how to treat a container or reservoir that have an open and closed state. The close overlap of terms suggests that synonyms for reservoir and container could be associated with the higher level term material suppliers without much confusion.

Table 6. Component View of Material Suppliers

% of pop	reservoir				container			
	#	name	% rep	% sum	#	name	% rep	% sum
5.9%	10	*tank*	*25.0%*	25.00%	4	*basket*	*11.1%*	11.11%
11.8%	9	*reservoir*	*22.5%*	47.50%	4	*carafe*	*11.1%*	22.22%
17.6%	3	bowl	*7.5%*	55.00%	4	*container*	*11.1%*	33.33%
23.5%	2	*carafe*	*5.0%*	60.00%	3	drum	*8.3%*	41.67%
29.4%	2	*container*	*5.0%*	65.00%	3	holder	*8.3%*	50.00%
35.3%	2	*cup*	*5.0%*	70.00%	3	*tank*	*8.3%*	58.33%
41.2%	2	*tray*	*5.0%*	75.00%	3	*tray*	*8.3%*	66.67%
47.1%	1	*basket*	*2.5%*	77.50%	2	compartment	*5.6%*	72.22%
52.9%	1	bin	*2.5%*	80.00%	2	hopper	*5.6%*	77.78%
58.8%	1	body	*2.5%*	82.50%	1	cartridge	*2.8%*	80.56%
64.7%	1	bucket	*2.5%*	85.00%	1	clip	*2.8%*	83.33%
70.6%	1	cover	*2.5%*	87.50%	1	*cup*	*2.8%*	86.11%
76.5%	1	housing	*2.5%*	90.00%	1	cylinder	*2.8%*	88.89%
82.4%	1	lid	*2.5%*	92.50%	1	pad	*2.8%*	91.67%
88.2%	1	pan	*2.5%*	95.00%	1	rack	*2.8%*	94.44%
94.1%	1	pitcher	*2.5%*	97.50%	1	*reservoir*	*2.8%*	97.22%
100.0%	1	plate	*2.5%*	100.00%	1	sponge	*2.8%*	100.00%

3.3.2 A Component Look at Guiders

Table 7 shows a detailed view of common names associated with the component naming term link. Again it is worth mentioning that 70% of all common name instances

are again realized within the first 30% of the population. A summary of all guider component terms is shown in Table 8. Unlike for material suppliers there is not a large amount of overlap between the varying terms.

Hinge, sled, link, and tube do have some instances of component overlap, but not as often as with material suppliers. It is worth noting that common names for bearing and diode are almost always the same as their respective component naming term. This may be because bearing and diode are both very specific, non-ambiguous components and are not often realized with varying types of form. The remaining guider terms carry a higher level of ambiguity as shown by their associated common names. Users have used the term sled to describe artifacts ranging from a car to a bolt, two artifacts that share no commonality.

Table 7. Detailed Component View of Link

link				
#	name	% pop	% rep	% sum
19	linkage	4.5%	22.9%	22.9%
12	arm	9.1%	14.5%	37.3%
9	link	13.6%	10.8%	48.2%
7	shaft	18.2%	8.4%	56.6%
6	bracket	22.7%	7.2%	63.9%
4	pin	27.3%	4.8%	68.7%
4	rod	31.8%	4.8%	73.5%
3	bar	36.4%	3.6%	77.1%
3	lever	40.9%	3.6%	80.7%
2	plate	45.5%	2.4%	83.1%
2	reciprocating piece	50.0%	2.4%	85.5%
2	tube	54.5%	2.4%	88.0%
1	cable	59.1%	1.2%	89.2%
1	clip	63.6%	1.2%	90.4%
1	extension	68.2%	1.2%	91.6%
1	fixture	72.7%	1.2%	92.8%
1	hammer	77.3%	1.2%	94.0%
1	lock	81.8%	1.2%	95.2%
1	sear	86.4%	1.2%	96.4%
1	slider	90.9%	1.2%	97.6%
1	wand	95.5%	1.2%	98.8%
1	wire	100.0%	1.2%	100.0%

Table 8. Component View of Guiders

#	name	Δ%	#	name	Δ%
bearing			diode		
17	bearing	94.44%	5	diode	83.33%
1	wheel	100.00%	1	bypass	100.00%
hinge			sled		
12	hinge	38.71%	8	car	40.00%
7	pin	61.29%	2	bracket	50.00%
5	rivet	77.42%	2	plate	60.00%
3	joint	87.10%	1	bolt	65.00%
…	…	…	1	button	70.00%
1	pivot	100.00%	…	…	…
link			1	train	100.00%
19	linkage	22.89%	tube		
12	arm	37.35%	31	tube	39.74%
9	link	48.19%	10	hose	52.56%
7	shaft	56.63%	8	pipe	62.82%
6	bracket	63.86%	5	line	69.23%
4	pin	68.67%	4	faucet	74.36%
4	rod	73.49%	…	…	…
…	…	…	1	window	100.00%
1	wire	100.00%			

4 RESULTS

Based on the functional and common naming analyses performed above, it is apparent that a hybrid approach is needed to formulate a natural language to component naming synonym method.

4.1 Combining Two Viewpoints

Neither common naming information or functional information alone suggest how to begin linking natural language synonyms. There are cases where an entire class of component naming terms overlap and others with hardly any overlap. Some component naming terms have overlapping common names and others have minimal overlap. For guiders, with the exception of the term diode, all components within a class appear to have a great deal of functional overlap, but minimal naming overlap. For material suppliers there is both functional similarity as well as common name overlap.

If synonyms are associated with the secondary level of the component naming taxonomy, there would be several cases where common names would not seem like they

belong together in the same list such as car and diode. If synonyms are to be associated with the component level of the component naming the question is how to appropriately address repeated common names. Should common names only be associated with a single component naming term, and if so what rules can be developed to determine where to assign a particular synonym? Looking back at the reservoir and container data users consistently use tray, carafe, and cup to describe both naming terms. Should tray be associated with container instead of reservoir because it has one more occurrence with container? Functionally reservoir and container are nearly identical and using either tray, cup, or tank would result in the same overall functional representation. Since the goal is to ultimately use the natural language terms to generate a functional model, duplication of common terms is necessary.

4.2 Proposed Method

In order to implement the hybrid strategy outlined above, a step by step method for gathering natural language terms from existing repository data is formulated here.

Step 1) Reduce verbose common names to their root by removing any unnecessary descriptors. For example "lower left ac cover" would become "cover."

Step 2) Generate a list of all common names and their rate of appearance associated with a given component naming term.

Step 3) Remove conflicting component naming terms that may exist within the list of common component names. A conflict occurs when a different component naming term appears in a list of common names for another component naming term. For example, the common name container (also a component naming term) appears in the listing for the naming term reservoir. For this case container would then be removed from the list of common names associated with a reservoir (shown in Table 9).

Step 4) Calculate the percent representation and a summation of percentages of each common name across the population and order from highest to lowest. An example is shown in Table 9.

Table 9. Reservoir Synonym Data Table

reservoir			
#	name	% represent.	% summation
10	tank	26.32%	26.32%
9	reservoir	23.68%	50.00%
3	bowl	7.89%	57.89%
2	carafe	5.26%	63.16%
2	cup	5.26%	68.42%
2	tray	5.26%	73.68%
1	basket	2.63%	76.32%
…	…	…	…
1	plate	2.63%	100.00%

Step 5) Accept all common names as natural language synonyms up to and including all terms required to reach a 70% threshold of the population. The highlighted terms from Table 9 would then be added as natural language synonyms for component naming term reservoir. Synonym terms listed in the component naming terms (shown in Section 2.2) will also be included in the overall set of natural language synonyms.

The 5 step algorithm shown will allow for synonym terms to be duplicated for different component naming terms. For example, the term tank would be listed as a synonym for both a reservoir and a container. If a user were to specify the component tank in FFF all artifacts tagged as a reservoir or container will be used for further functional analysis. As shown in Section 3.2 it is expected that overlapping terms will have similar if not identical functionality.

5 COMPONENT NAMING SYNONYMS

This section presents the natural language component synonyms (Table 10) found using the method shown in Section 4.2. Only component basis terms that have occurred in the repository are shown.

Table 10. Natural Language Synonyms

Component Basis Term	Natural Language Synonyms
abrasive	sand paper, conductor, switch, sensor
acoustic insulator	muffler
agitator	stirrer
airfoil	airfoil, wing
battery	battery
bearing	bearing
belt	belt, webbing, zip tie
blade	blade
bracket	bracket
brush	brush, bristle, ring
burner	burner, sparkler
cam	cam, counterweight, converter
cap	cap, plug
capacitor	capacitor
carousel	carousel, turntable, scraper
choke	coil
circuit board	chip, control board
clamp	clamp, gripper, caliper, chuck, clip, collet, crimp
clutch	clutch, actuator, spacer
condenser	condenser
container	basket, carafe, container, holder, tank, drum, tray, hopper, compartment
cover	cover, plate, lid
crank	crank
cushion	pad, cushion, panel, dampener, bushing, foam
diode	diode
divider	spacer, divider, plate
door	door, window
electric conductor	crimp, conductor, connector, power cord, contact
electric cord	plug and cord, power cord
electric distributor	bus
electric insulator	insulator, backing material
electric motor	motor
electric plug	plug and cord, plug, inner connection
electric resistor	resistor
electric socket	socket, input, outlet, jack
electric switch	switch, button
em sensor	antenna, bar
extension	holder
fan	fan, impeller

Table 10. Natural Language Synonyms (cont.)

Component Basis Term	Natural Language Synonyms
fastener	fastener, bolt, weld, nut
flywheel	bobbin, holder, hammer
friction enhancer	grip, pad, feet, filter, clip, tape, roller, sticker
fuse	fuse
generator	alternator
handle	handle
heat exchanger	mass, peltier device, chamber, radiator, pipe
heating element	heater
hinge	hinge, pin, rivet, joint
housing	housing, case, shell
hydraulic piston	piston, bolt, plunger
hydraulic pump	cylinder, pump
ic motor	engine
inclined plane	Slope
inductor	inductor, transformer
insert	liner, pin, drop forward, connector, insert, bushing, die
knob	knob, button, pad
latch release	clip, button, lock, release, brake, holder, pin, switch
lens	lens
lever	lever, pedal, trigger, actuator, control lever
light source	lamp, bulb
link	linkage, link, rod, bar, pin, arm
magnet	magnet, rails
material filter	filter, bag, grate pre-filter
mechanical transformer	transformer, dashboard
needle	needle, mount
nozzle	nozzle, shower head, neck, guard
pneumatic piston	piston, booster
pneumatic pump	pump
pressure gauge	gauge
projectile	ball
punch	punch
reservoir	tank, reservoir, bowl, carafe, cup, tray
rotational coupler	coupler, rotor, input, coupling
screw propeller	impeller
seal	gasket, o-ring, seal
shaft	shaft, axle, driveshaft
sled	car, bracket, plate, bolt, button

Table 10. Natural Language Synonyms (cont.)

Component Basis Term	Natural Language Synonyms
speaker	speaker
spring	spring
stop	stop, snap ring, lock washer, stopper, feet, sphere, gear lock, bumper, visor
stuffing	foam
support	plate, support, ring, mount, base, pin, clip, holder, bar, frame, guide, foot, spacer, body
thermal conductor	plate
thermal insulator	shield, ceramic, fiberglass, insulator
thermostat	thermostat, sensor
transistor	junction, chip, solid state silver bond, transistor
tube	tube, hose, pipe, line, faucet
valve	valve, stopper, flap, sieve
visual indicator	gauge, guide, plate, ball
washer	washer
wheel	wheel, rotor

6 CONCLUSIONS AND FUTURE WORK

This work establishes the natural language interpretation foundation necessary to support the envisioned Form Follows Form method. Natural language interpretation of components is essential to allow novice engineers and designers to specify an initial product concept that, once interpreted, can be parsed and used as input for existing concept generation algorithms. This ability, paired with the emerging Form Follows Form method, is anticipated to make design more accessible to the larger engineering community by removing the need to be well versed in naming taxonomies

The natural language to component naming terms method presented establishes an approach that imbues a machine with the ability to learn the association between human speak and the standardized set of component naming terms as the knowledge base in the Repository. An initial set of natural language to component naming terms is generated by the AI method for the current state of the Design Repository and

presented. After review of the results, we observe that the normative nature of the Design Repository (i.e., entry by many different contributors with varying descriptive styles) indeed captures a wide array of natural language terms that support the interpretation algorithm.

Future work includes the task of monitoring common names associated with artifacts as additional products are cataloged. There may be the need to update or modify the parsing algorithm as more products are added to the repository systems. This work also establishes a framework for analyzing the component naming taxonomy. As shown in Section 5 there are several component naming terms that have not yet been used to represent a single artifact. Further analysis may find that the component basis taxonomy naming terms could be removed or modified.

It is possible that additional natural language synonyms could be found using alternate sources. Additional synonyms could possibly be found by searching engineering catalogs, patents, and texts. This process would also help to verify the natural language synonyms that have already been identified in this work. Natural language synonyms could also be realized and verified by using language sources such as WordNet (http://wordnet.princeton.edu/). WordNet may best aid by adding natural language synonyms to component basis terms not yet realized within the design repository.

REFERENCES

[1] Otto, K., and Wood, K., 2001, Product Design: Techniques in Reverse Engineering, Systematic Design, and New Product Development, Prentice-Hall, New York.

[2] Pahl, G., and Beitz, W., 1996, Engineering Design: A Systematic Approach, Springer Verlag.

[3] Ullman, D. G., 2002, The Mechanical Design Process 3rd Edition, McGraw-Hill, Inc., New York.

[4] Ulrich, K. T., and Eppinger, S. D., 2004, Product Design and Development, McGraw-Hill/Irwin, Boston, MA.

[5] Bryant, C., Bohm, M., Stone, R., and Mcadams, D., 2007, "An Interactive Computational Design Tool: A Hybrid of Two Methods," Proc. Proceedings of the IDETC/CIE 2007, Las Vegas, NV, DETC2007-35583.

[6] Bryant, C., Stone, R., Greer, J. L., Mcadams, D., Kurtoglu, T., and Campbell, M., 2007, "A Function-Based Component Ontology for Systems Design," Proc. International Conference on Engineering Design, ICED'07, Paris, France, Paper 643.

[7] Kurtoglu, T., Campbell, M., Bryant, C., Stone, R., and Mcadams, D., 2008, "A Component Taxonomy as a Framework for Computational Design Synthesis," Journal of Computers and Information Science in Engineering, Accepted, to appear(

[8] Kurtoglu, T., and Campbell, M., 2007, "Exploring the Worth of Automatically Generated Design Alternatives Based on Designer Preferences," Paris, France.

[9] Bryant, C., Mcadams, D., Stone, R., Kurtoglu, T., and Campbell, M., 2005, "A Computational Technique for Concept Generation," Proc. Proceedings of IDETC/CIE 2005, Long Beach, CA, DETC2005-85323.

[10] Bryant, C., Stone, R., Mcadams, D., Kurtoglu, T., and Campbell, M., 2005, "Concept Generation from the Functional Basis of Design," Proc. International Conference on Engineering Design, ICED 05, Melbourne, Australia.

[11] Kurtoglu, T., and Campbell, M., 2008, "Automated Synthesis of Electromechanical Design Configurations from Empirical Analysis of Function to Form Mapping," Journal of Engineering Design, Accepted, To Appear (http://www.informaworld.com/10.1080/09544820701546165).

[12] Kurtoglu, T., Campbell, M., Gonzalez, J., Bryant, C., Stone, R., and Mcadams, D., 2005, "Capturing Empirically Derived Design Knowledge for Creating Conceptual Design Configurations," Proc. Proceedings of IDETC/CIE 2005, Long Beach, CA, DETC2005-84405.

[13] Russell, S. J., and Norvig, P., 2003, Artificial Intelligence: A Modern Approach (2nd Ed.), Prentice Hall, Upper Saddle River, NJ.

[14] Bohm, M., Stone, R., and Szykman, S., 2005, "Enhancing Virtual Product Representations for Advanced Design Repository Systems," Journal of Computer and Information Science in Engineering, 5(4), pp. 360-372.

[15] Szykman, S., Sriram, R., and Regli, W., 2001, "The Role of Knowledge in Next-Generation Product Development Systems," Journal of Computer and Information Science in Engineering, 1(1), pp. 3-11.

[16] Szykman, S., Fenves, S., Shooter, S., and Keirouz, W., 2001, "A Foundation for Interoperability in the Next-Generation Product Development Systems," Computer-Aided Design, 33(7), pp. 545-559.

[17] Bohm, M., Stone, R., Simpson, T., and Steva, E., 2008, "Introduction of a Data Schema: To Support a Design Repository," Computer-Aided Design, 40(7), pp. 801-811.

[18] Shooter, S., Simpson, T., Kumara, S., Stone, R., and Terpenny, J., 2005, "Toward a Multi-Agent Information Management Infrastructure for Product Family Planning and Mass Customisation," International Journal for Mass Customisation, 1(1), pp. 134-155.

[19] Bohm, M., Vuchovich, J., and Stone, R., 2007, "An Open Source Application for Archiving Product Design Information," Proc. Proceedings of DETC'07, Las Vegas, NV, DETC2007-35401.

[20] Chiu, I., and Shu, L., 2007, "Biomimetic Design through Natural Language Analysis to Facilitate Cross-Domain Information Retrieval," Artificial Intelligence for Engineering Design, Analysis & Manufacturing, 21(pp. 45-59.)

[21] Chiu, I., and Shu, L., 2007, "Using Language as a Related Stimuli for Concept Generation," Artificial Intelligence for Engineering Design, Analysis & Manufacturing, 21(pp. 103-121.)

[22] Chakrabarti, A., Sarkar, P., Leelavathamma, B., and Nataraju, B. S., 2005, "A Functional Representation for Aiding Biomimetic and Artificial Inspiration of New Ideas," Artificial Intelligence for Engineering Design, Analysis & Manufacturing, 19(pp. 113-132.)

[23] Luc, D. T., 2008, Pareto Optimality, Game Theory and Equilibria, Springer New York, Chap. 481-515.

[24] Deb, K., 2001, Multi-Objective Optimization Using Evolutionary Algorithms, Wiley and Sons, New York.

5. FORM FOLLOWS FORM – IS A NEW PARADIGM NEEDED?

Matt R. Bohm
Robert L. Nagel
Robert B. Stone, Ph.D.
Design Engineering Lab
Department of Interdisciplinary Engineering
Missouri University of Science and Technology
Rolla, MO 65409

ABSTRACT

This paper presents a new form-based concept generation technique known as Form Follows Form (FFF). The technique allows a novice engineer or designer to use natural language to specify components envisioned within a product to initiate a more thorough concept generation process. Form follows form takes the initial component solution and then formulates the underlying function structure by leveraging a repository of over 5500 artifacts. Existing computational conceptual design methods are then employed to automatically display a set of ranked concept alternatives to the user. Users can choose from two different levels of interaction, an automatic mode that uses the most common functions to develop concept alternatives, or a mode that allows the user to be more precise in defining a product's interaction. The computational algorithms and grammar rules are detailed along with a case study using both tiers of interaction.

1 INTRODUCTION

Today the United States is the leader in technology innovation. That innovation or creativity results from funded national initiatives (e.g. NSF, DoD), from large companies supporting their own research to stay competitive (e.g. Intel, 3M, Boeing, Northrop Grumman, Apple) and from smaller companies driven to address a perceived market need. The nation's standing as an innovation leader is now more tenuous than it

has been in the past half-century due to a number of factors ranging from the current economic downturn to the effects of globalization and emerging economic forces in formerly third-world countries. Now, more than ever, methods that support innovation need to be studied to ensure quality of life continues as expected. Innovative concept generation is still widely viewed as a magical quality largely not characterized by scientific phenomena. Ideas about fostering innovation in product design have been rampant in psychological and design literature in the last twenty years, but no definitive studies have emerged to prescribe practices that positively impact creativity or innovation.

Collaborative research projects with national labs, defense agencies and industry coupled with teaching engineering students, we have observed that most often engineers and designers think in terms of components. They visualize the physical implementation instead of interpreting the functional requirements. For example, engineers in NASA JPL's Team X select from standard space systems when developing a mission and its subsequent spacecraft, even though unknown, new, emerging systems may ultimately be implemented on the actual spacecraft. When GM's design for six-sigma engineers try to associate performance equations with customer demand through automotive sub-system functionality, often the functional description created contains a mix of components and pseudo-functions and is not necessarily computable using existing methods .

This paper reports on efforts to 1) capture components within an envisioned concept or design, 2) infer user intent of the designated components, and 3) apply a set of grammar rules to output a resulting functional model. Section 3 presents technical details of capturing concept information and the set of grammar rules used for functional model generation. Section 4 begins by demonstrating functional model generation using FFF using two different levels of user involvement. Finally, Section 4 compares a human generated functional model to the FFF counterpart.

2 BACKGROUND

In this section we present an overview of current and emerging concept generation techniques. We begin in Section 2.1 by reviewing techniques associated with manual concept generation activities. Sections 2.2 and 2.3 outline more recent efforts to apply artificial intelligence (AI) techniques to concept generation. Finally, a brief overview of a design repository is presented in Section 2.4.

2.1 Concept Generation Techniques

A variety of concept generation methods exist for application to engineering design problems – from those that are common practice within the field of design to the more modern computer aided concept generation methods. Many researchers have sought to formalize the conceptual design phase. Antonsson and Cagan concisely define the notion of 'formal' as "...computable, structured, and rigorous, not ad hoc" [1]. Furthermore, by founding concept generation techniques on functionality, solution-independent design descriptions can be built [2]. Such methods generally rely on a form of functional decomposition of the overall problem to initiate the search for physical design solutions during conceptual design. Whether driven in this function-based manner or otherwise, much variability is exhibited in just how this search is carried out depending on the method chosen. This reflects the variety of perspectives that have been suggested for addressing the conceptual design problem and a sampling of the major themes is reviewed next.

2.1.1 C-Sketch/6-3-5 Method

The 6-3-5 method is a generic technique that supports innovative thinking [3]. In 6-3-5, members of an engineering design team (optimally 6-8 members) generate, interpret, and modify the individual ideas of other team members by first brainstorming and sketching individually on three ideas for various aspects of the product, then passing their ideas to the next team-member who adds additional ideas and sketches. C-Sketch is a variant of the 6-3-5 method wherein members produce only sketches and refrain from communicating verbally when passing ideas to the next member [4].

Passing only sketches allows other team members the opportunity to interpret the concepts in a different manner than the original author, thereby increasing design diversity.

2.1.2 The Catalog Design Method

Another approach, referred to as catalog design, is based on a catalog of physical elements (components, assemblies, etc.) that can be browsed for solutions that match required performance specifications. The data for design catalogs are limited to some degree insofar as these design catalogs are generally a subset of previously designed systems, which leads to the issue of potential novelty restrictions. However, a major benefit of catalog design is the ability to utilize design knowledge that falls outside human memory [5-7]

2.1.3 Design by Analogy

In Design by Analogy, a functional model is created of the product being designed. Examining analogous products or components that perform the same function generates solutions to the present design problem. The designer then evaluates these similar components for appropriateness in solving the given design problem [8]. One Design by Analogy method widely recognized in the engineering design community is the Theory of Inventive Problem Solving, or simply TRIZ. TRIZ was developed by Altshuller during the 1940-50's period and was based on the examination of large numbers of existing patents [9]. The end result of this effort is an engineering design approach that identifies a set of conflicts that occur in design along with a set of principles that can be applied to generate solutions that solve these conflicts.

2.1.4 Morphological Matrix Method

The morphological matrix introduced by Zwicky is now a classic technique for use in conceptual design [10]. This method provides the design engineer with a simple, albeit manual, means for bookkeeping potential physical solutions and their corresponding functionality.

2.2 Foundations in Automated Concept Generation

The front end of the conceptual design process has seen few attempts at automation, perhaps due in part to the evolving strategies and methodologies that exist for this phase of design. However, over the past decade, several methodologies have coalesced around the functional decomposition and partial solution manipulation techniques originally introduced by Pahl and Beitz [11], e.g., [12-21]. These methodologies take a designer through a set of steps to help decompose a design problem and build conceptual solutions based on the functionality that a product needs to exhibit. Function modeling methods abstract the functionality that a solution must fulfill from the established customer needs, ideally removing designer biases that may be introduced by focusing on specific solutions too early in the design process. This abstraction helps a designer generate more complete conceptual solutions and balance design choices between different components with the same functionality [11].

Research into the benefits of structured design methods (e.g., [22]) coupled with research into designers' reluctance to use them (e.g., [23, 24]) seem to point toward the need for the seemingly tedious stages of systematic design to employ some level of automation to help integrate the benefits of a structured method with the more natural activities of a designer – a need that is most evident during the early phases of conceptual development.

Computational tools for conceptual design do exist, yet these tools often address areas that support aspects such as initial requirements gathering (e.g., organizational tools such as the TikiWiki project [25], the creation of function structures (e.g., the function grammar tool developed by Sridharan and Campbell [26]), or optimization of well-established concepts (e.g., [27]) rather than the translation of functional requirements into creative solutions).

2.3 The State of the Art in Automated Concept Generation

Computerized concept generation techniques, spanning the broad AI topics of knowledge representation and reasoning, promise engineers a faster realization of potential design solutions based upon previously known products and implementations. While the area of automated concept generation has made great strides in recent years, most methods still require the user to indicate desired functionality. Using functional descriptions has been shown to help engineers stray away from pre-trained ideas of how a product or device would look and operate, although can cause confusion for engineers and scientists who have not been trained to describe product functionality. Two of the automated concept generation methods under development today rely solely on the user's ability to develop functional descriptions of their desired product. Both of these methods make use of a repository of design information including component connection information and component functionality.

The recent foundations for concept generation through computational reasoning have been developed based on formalisms for describing function or purpose in engineering design largely led by members of our research team [28, 29]. Some of the results of this research include the development of a design repository to allow designers to store and retrieve design knowledge at various levels of abstraction, from form (components, sub-assemblies and assemblies) to configurations to function. Offering a fully functional and intuitive way to record product design information has been key to the acceptance of repositories as an important concept generation tool for designers. A prototype design repository framework by NIST guided the design repository (discussed further in Section 2.4) project to a more mature state.

The bank of empirical knowledge relating components to functions leads to the development of relational matrices [30, 31] and graph grammar rules [32, 33] that, when combined with a search mechanism, automatically creates conceptual designs. Aiding the methods set forth by Bryant and Kurtoglu [34, 35] is a component naming taxonomy

spanning 140 different component classifications. With the open-endedness or large degree of variability in conceptual design, numerous solutions are created through the search mechanisms (on the order of thousands). Presenting these thousands of solutions to the user is similar to an Internet search that produces thousands of results. It is overwhelming to the user and impractical to expect that such a large number of alternatives will be useful to the designer. Furthermore, the results showed that subtle challenges in a given design problem may not always be captured in the specification of initial function, and thus many results were not relevant to the user's needs [36, 37]. As a result, the proof of concept Designer Preference Modeler [38, 39] was created to find within the large set of results which concepts were most meaningful to the designer. By ranking select concepts, the search mechanism learns what aspects of the concept the user prefers, and seeks solutions that maximize the predicted preference. Initial results for this method are promising, but the impact they have on the design process is still unclear.

2.4 The Design Repository

The objective of a Design Repository is to allow designers to store and retrieve design knowledge at various levels of abstraction, from form (components, sub-assemblies and assemblies) to architecture description to function. Currently the Missouri S&T Design Repository contains design information for over 125 consumer based electro-mechanical products. Design information captured by the repository can be divided into seven main categories including: artifact-, function-, failure-, physical-, performance-, sensory- and media-related information types. The different levels of abstraction and types of design information provide innovative ways to approach design. With a well populated repository, emerging concept generator algorithms take, as input, basic product functionality or component information and instantaneously develop, filter and rank concepts to use as baselines for further product development.

While the possibilities design repositories offer are diverse and helpful to designers, the implementation of such repositories are crucial to their overall success and usefulness.

Realizing the potential impact of an operational Design Repository, researchers at Missouri S&T, The University of Texas at Austin and the National Institute of Standards and Technology (NIST) began gathering artifact information in 1999 [40-42]. Since that time, the process in which artifact data is gathered and recorded has changed significantly. Initially, artifact design information was recorded in spreadsheets and mainly took the form of Bills of Materials (BOM), Function Component Matrices (FCM), and Design Structure Matrices (DSM). While this type of information was useful, it was also limited in scope and the required matrix multiplications were quite cumbersome. A prior Design Repository initiative by NIST helped to guide the Design Repository project at Missouri S&T to a more mature state. To enhance data integrity, design information was migrated from spreadsheet form to a relational database. A web-based repository navigator including search and design tool generation features was created along with a repository entry application.

More recently, Missouri S&T has further partnered with UT-Austin [30, 31], Penn State [43], Virginia Tech, Bucknell [44], University of Buffalo and Texas A&M to expand the types of design information and breadth of design tool features within the repository. The Design Repository serves as a hub for designers for information exchange and design generation tools and is heavily utilized in the current VOICE project. Information entry and retrieval occurs within a standalone application [45] (available at http://designengineeringlab.org/repositoryEntry/) while information retrieval occurs over the Internet through the Design Repository's web portal (http://repository.designengineeringlab.org/). The infrastructure supporting these two applications is the Design Repository database and schema [43]. The database schema establishes what types of design information can be stored, the relationship of those

elements and the extensibility of including new and additional types of design information.

3 RESEARCH APPROACH

In order to explore the possibility of a form-initiated, AI-enabled concept generation paradigm, initial steps require a systematic approach to abstracting functional descriptions from an initial form-based concept seed. From there, the AI generated functional model can be used as input to the existing concept generation algorithms (from Section 2.3). The overall research approach followed is decomposed into three specific activities.

1) Capture chains of envisioned components for a given concept by using computer parse-able natural-language component terms;
2) Capture designer preferences to determine intent of the concept; and
3) Explore AI reasoning approaches to derive a functional representation of the concept.

3.1 Objective 1 – Capturing Chains of Components

Users are allowed to specify an initial solution by listing chains of components envisioned in their product by using an augmented component naming taxonomy [46]. This is a task that initially appears to be simple, however, there are several ways one could describe information about components such as a DSM or a simple listing of components. Since ease of use and accessibility are key drivers of this research, asking a user to first generate a component connection matrix would be a cumbersome task. Alternatively, a simple list of components may not effectively capture the intent of the user (that is component connection and ordering). From a computational standpoint, information regarding components needs to, at a minimum, infer how the user intends those components to be connected to one another. For example, a user lists out wire, shaft and motor as components in a concept. A basic search of the design repository would show those components have not been observed to connect in that particular order. It is therefore necessary to build a framework that allows for an easy and logical

manner to gain information about components in a particular concept. With a semi-logical ordering of input components, algorithms will be better positioned to statistically determine the intended component order.

Figure 1 shows the proposed approach to capture a perceived concept. The example shown in Figure 1 contains the basic components of an ice tea maker and will be used throughout the remainder of this paper. Users are asked to enter components as a series of discrete chains. As shown, the first chain of components includes a cord, switch, and a heating element while another may consist of a tank, tube and a condenser. Once chains are entered a user is allowed edit, remove or reorder specific chains or components.

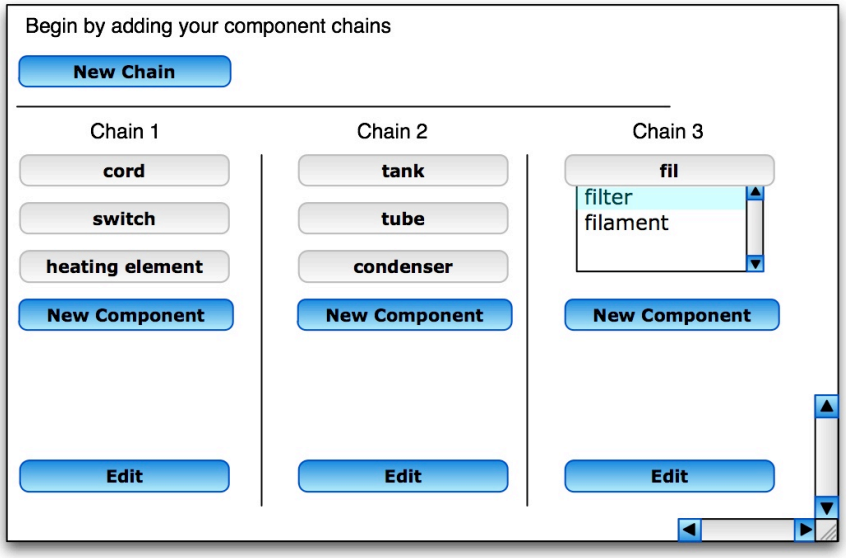


Figure 1. Component capture screenshot

3.2 Objective 2 – Determining User Intent

At this stage designer preferences are needed in order to provide additional information such that computational reasoning about the intended use of their product may proceed. Once the component chains are designated by the user it will be necessary

to systematically determine the intended use of particular components. Data from the repository will be utilized to determine which functions and flows are solved by each of the given input components. Most artifacts in the repository are given a common name as well as a more accurate component basis taxonomy name [34, 35]. For example, a user might list "small dc motor" as a common name, but also choose "motor" from the component naming taxonomy. The component naming taxonomy exists to remove ambiguity from common names and to aid in the clustering of design information. Across the entire repository each component naming term is associated with, on average, 17.7 unique function-flow pairs. This non-exclusive relationship between function flow pairs and components occurs because some components solve more than one function for a particular implementation and some components have multiple distinct uses. It is therefore necessary to determine which function(s) and flow(s) are intended by the user's selection of a particular component.

To determine user intent we present a two-tiered approach for user involvement: Tier 1 – algorithms automatically select the most prevalent chain of functions associated with a given component, or Tier 2 – the user is prompted to identify the primary flow (material, energy or signal) traversing a particular component. As an example for both tiers of user involvement we present a simplified functional model (Figure 2) of an ice tea maker. For brevity the model ignores thermal sensing, on/off switches and interaction. A functional model of this type is not always common to engineers, but what is common is the identification of components. Components have been associated with individual or groupings of functions within the model for clarity. For example, the functions of importing and transferring electrical energy have previously been observed to solved by the component cord whereas the functions of converting electrical energy (EE) to thermal energy (TE), transferring TE and converting liquid to gas have been solved by a heating element.

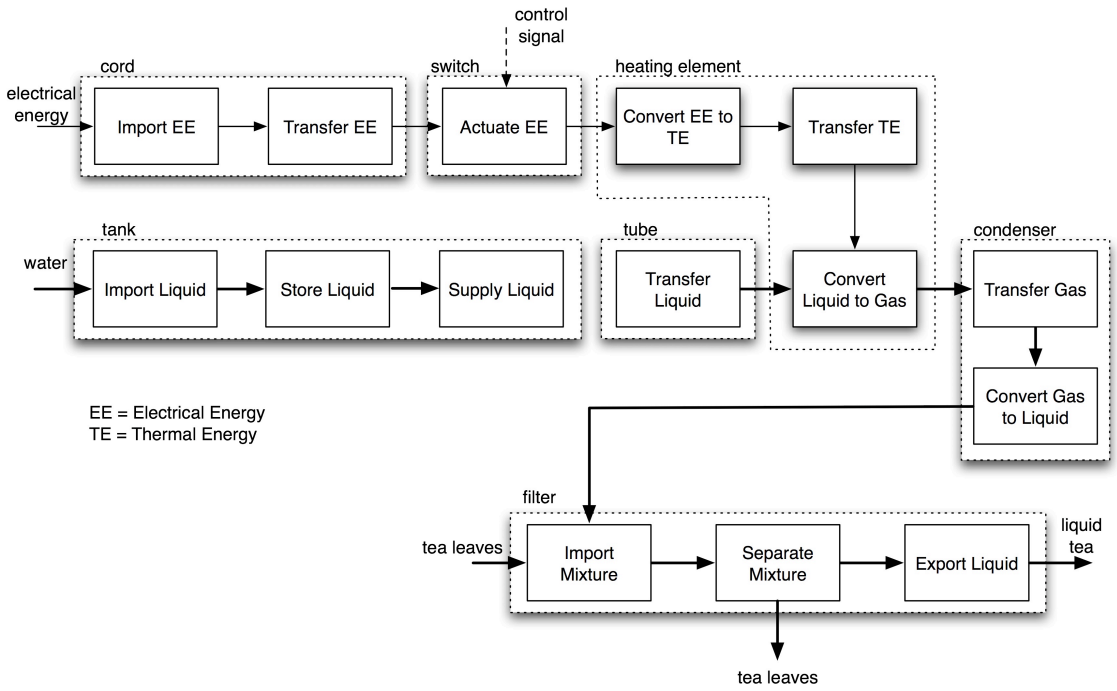


Figure 2. Simplified functional model of an ice tea maker

Examination of repository data shows that in general 70% of both functions and flows are realized within the first 30% of unique instances of a particular component. This finding suggests that the 70/30 allocation is Pareto optimal [57]. Table 1 shows the top 70% of instances of both function and flow for each of these components designated in the functional model in Figure 3. For all of the components, except for tank, no component synonym lookup or natural language exploration has been conducted at this stage. Table 1 along with the functional model in Figure 2 is used for illustration as the proposed two-level tiered approach is presented next.

3.2.1 Tier 1 Approach

This tier is analogous to Google's "I'm Feeling Lucky" search option. The algorithm will query the database and determine all of the top rated functions and flows associated with each of the components of the ice tea maker. Looking specifically at the component tank, the highest ranked functions are export, import, store, stop, and

position with the highest-ranking flow being a solid material. At this point the system will designate the functionality of the tank as importing, storing, stopping, provisioning and exporting a solid material. In order to better refine the output order of functionality a series of grammars will be formulated (Section 3.3) and applied such that the selected functions are organized in a logical fashion. Previous efforts [26, 32, 47] have sought such functional model creation grammars and those efforts will be leveraged and extended. For example, from a modeling sense it is not logical to export something before it is imported.

Table 1. Ice tea maker components and their associated functions and flows

	component	function	delta %	running %	flow	delta %	running %
chain 1	cord	transfer	44.2%	44.2%	electrical	98.7%	98.7%
		import	39.0%	83.1%			
	switch	actuate	33.7%	33.7%	electrical	31.1%	31.1%
		import	19.6%	53.3%	human energy	23.1%	54.3%
		convert	17.7%	71.0%	control	19.3%	73.6%
	heating element	transfer	35.4%	35.4%	electrical	43.9%	43.9%
		convert	24.4%	59.8%	thermal	18.3%	62.2%
		change	9.8%	69.5%	solid	13.4%	75.6%
chain 2	tank	export	23.4%	23.4%	liquid	25.0%	8.3%
		import	17.6%	41.0%	solid	22.1%	16.7%
		store	13.5%	54.5%	thermal	15.6%	25.0%
		stop	12.7%	67.2%	mixture	15.2%	33.3%
		position	4.9%	72.1%	mechanical	8.2%	41.7%
					gas	5.7%	50.0%
					hydraulic	3.3%	58.3%
					human material	1.6%	66.7%
					chemical	1.2%	75.0%
	tube	import	22.3%	22.3%	solid	27.9%	27.9%
		export	21.7%	44.0%	mixture	15.2%	43.1%
		transfer	19.4%	63.3%	liquid	11.7%	54.8%
		guide	11.1%	74.5%	thermal	11.7%	66.6%
					gas	8.8%	75.4%
	condenser	transfer	66.7%	66.7%	gas	66.7%	66.7%
		convert	33.3%	100.0%	liquid	33.3%	100.0%
chain 3	filter	separate	25.9%	25.9%	solid	31.5%	31.5%
		export	16.7%	42.6%	mixture	25.9%	57.4%
		import	16.7%	59.3%	gas	20.4%	77.8%
		store	9.3%	68.5%			
		stop	7.4%	75.9%			

3.2.2 Tier 2 Approach

The second tier is analogous to a Google search aimed at a specific Internet address. The fundamentals of this type of user involvement are similar to the Tier 1 approach, however, the user will be asked to enter information regarding the primary flow (material, energy or signal) moving through a particular chain of components. The approach differs from the example presented in Section 3.2.1 in that the user would be allowed to select the dominant flow passing through a particular component. In the tier 1 approach the system automatically selects that a solid material is passing through the tank. In actuality the purpose of the tank is to move liquid through the system. Tier 2 will assign the same functionality but allow for the user to select the primary flow of liquid. Again, rules and grammars will create a logical ordering of functionality as well as associate natural language synonyms to standard flow terms.

3.3 Objective 3 – Reasoning to Derive a Functional Representation

Reasoning is one of the major topics of research within the AI community [48-50]. It is applicable to the current concept generation problem of transforming an initial seed concept into a more abstract representation of underlying functionality to initiate existing automated search and synthesis algorithms. Prior research has produced grammar rules that generate function structures from overall input and outputs of a product [26, 47] and to transform functions to components (that solve the functionality) [32], but none exist to go from components to functions.

Grammar rules are associated with individual functions and dictate the allowed incoming and outgoing flows. A set of rules has been developed for each tier. Tier 1 one requires no user interaction and relies on grammar rules to assign incoming/outgoing flows. Tier 2 allows for user interaction, more specifically the user is asked to designate a flow at the input of a function chain, when multiple flows are associated with a function, or when a function definition states that an output flow must be different from an output flow. Table 2 summarizes the grammar rules for each function and shows a

graphical example of a function operating on Energy, Material or Signal (EMS) flow(s) for different classes of rules. Previous research has concluded that the secondary level of the functional basis is sufficient for most types of representation [51], as such the grammar rules are only associated with the secondary level of the functional basis.

In addition to the grammar rules, FFF will also make use of the following definitions and global rule for each tier: *Continuing Flow* – A continuing flow is a flow that is both the output of the previous function and the input to the next function. *Dangling Flow* – A dangling flow is a flow that is connected to a single function (incoming or outgoing) but does not continue to or originate from another function within the functional model. As a global rule, no functions may be duplicated sequentially.

Table 2. Grammar Rules for Form Follows Form

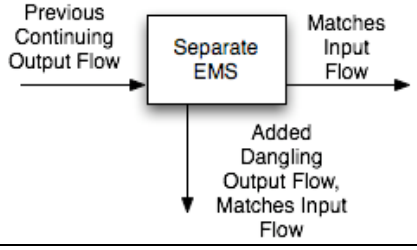
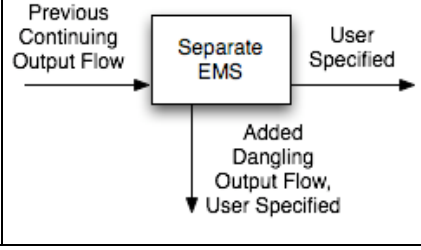
	Tier 1		**Tier 2**	
Function	*Incoming Flow(s)*	*Outgoing Flow(s)*	*Incoming Flow(s)*	*Outgoing Flow(s)*
Separate	Uses previous flow from function chain.	Continuing output flow matches input flow. Additional dangling output flow created matching continuing output flow.	Uses previous flow from function chain.	User allowed to select continuing output flow as well terminating output flow from component flow list.
				

Table 2. Grammar Rules for Form Follows Form (cont.)

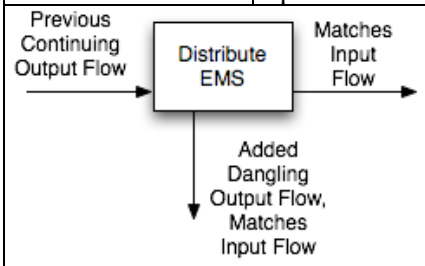
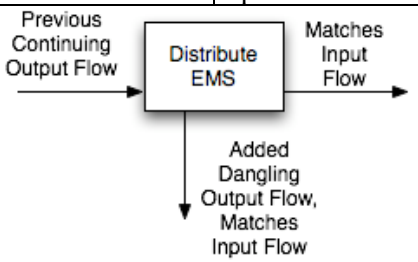
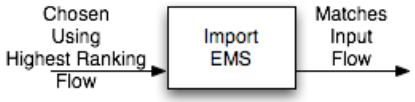
<table>
<tr><th></th><th colspan="2">Tier 1</th><th colspan="2">Tier 2</th></tr>
<tr><th>Function</th><th>Incoming Flow(s)</th><th>Outgoing Flow(s)</th><th>Incoming Flow(s)</th><th>Outgoing Flow(s)</th></tr>
<tr><td rowspan="2">Distribute</td><td>Uses previous flow from function chain.</td><td>Continuing output flow matches input flow. Additional termination output flow created using input flow.</td><td>Uses previous flow from function chain.</td><td>Continuing output flow matches input flow. Additional termination output flow created using input flow.</td></tr>
<tr><td colspan="2"> 
</td><td colspan="2"> 
</td></tr>
<tr><td rowspan="2">Import</td><td>Automatically placed as the first function for a chain of components. Highest ranking flow chosen as input.</td><td>Output matches input.</td><td>Automatically placed as the first function for a chain of components. User allowed to select input flow from component associated flows.</td><td>Output matches input.</td></tr>
<tr><td colspan="2"> 
</td><td colspan="2"> 
</td></tr>
<tr><td>Export</td><td>Automatically placed as the last function for a chain of components. Uses previous flow from function chain.</td><td>Output matches input. Final function in a chain of functions.</td><td>Automatically placed as the last function for a chain of components. Uses previous flow from function chain.</td><td>Output matches input. Final function in a chain of functions.</td></tr>
<tr><td>Transfer</td><td>Uses previous flow from function chain.</td><td>Output matches input.</td><td>Uses previous flow from function chain.</td><td>Output matches input.</td></tr>
<tr><td>Guide</td><td>Uses previous flow from function chain.</td><td>Output matches input.</td><td>Uses previous flow from function chain.</td><td>Output matches input.</td></tr>
</table>

Table 2. Grammar Rules for Form Follows Form (cont.)

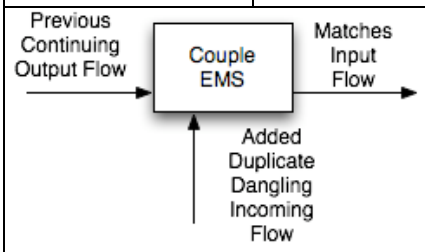
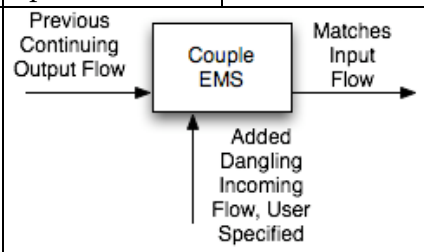
	Tier 1		Tier 2	
Function	*Incoming Flow(s)*	*Outgoing Flow(s)*	*Incoming Flow(s)*	*Outgoing Flow(s)*
Couple	Uses previous flow from function chain as the continuing flow. Duplicate input flow automatically created.	Single output flow matches original input flow.	Uses previous flow from function chain as the 1st input flow. Prompts the user to designate a 2nd input flow.	User allowed to select output flow.
				
Mix	Uses previous flow from function chain as the continuing flow. Duplicate input flow automatically created.	Single output flow matches original input flow.	Uses previous flow from function chain as the 1st input flow. Prompts the user to designate a 2nd input flow.	User allowed to select output flow.
Actuate	Uses previous flow from function chain as the continuing flow. Additional input control signal flow created.	Output matches input.	Uses previous flow from function chain as the continuing flow. Additional input control signal flow created.	Output matches input.
Regulate	Uses previous flow from function chain as the continuing flow. Additional input control signal flow created.	Output matches input.	Uses previous flow from function chain as the continuing flow. Additional input control signal flow created.	Output matches input.
Change	Uses previous flow from function chain.	Output matches input.	Uses previous flow from function chain.	Output matches input.

Table 2. Grammar Rules for Form Follows Form (cont.)

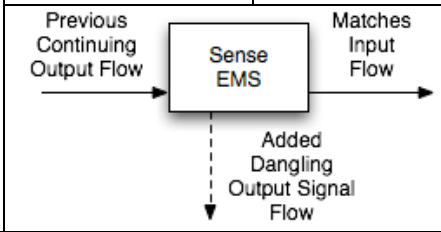
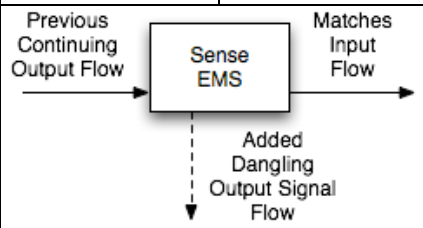
	Tier 1		Tier 2	
Function	*Incoming Flow(s)*	*Outgoing Flow(s)*	*Incoming Flow(s)*	*Outgoing Flow(s)*
Stop	Uses previous flow from function chain.	Output matches input.	Uses previous flow from function chain.	Output matches input.
Convert	Uses previous flow from function chain.	Highest ranking flow for a given component flow list that does not match the input flow.	Uses previous flow from function chain.	User chooses new outgoing flow from flow list. If selected output matches input then function is removed.
Store	Uses previous flow from function chain. Must be directly followed by the Supply Function.	Output matches input. Additional function of supply automatically attached.	Uses previous flow from function chain. Must be directly followed by the Supply Function.	Output matches input. Additional function of supply automatically attached.
Supply	Uses previous flow from function chain. Must be directly proceeded by the Store Function.	Output matches input.	Uses previous flow from function chain. Must be directly proceeded by the Store Function.	Output matches input.
Sense	Uses previous flow from function chain as continuing flow.	Output matches input. Additional terminating status signal flow automatically attached.	Uses previous flow from function chain as continuing flow.	Output matches input. Additional terminating status signal flow automatically attached.
				
Indicate	Must be a status signal flow.	Output matches input.	Must be a status signal flow.	Output matches input.
Process	Must be a status signal or control signal flow.	Output matches input.	Must be a status signal or control signal flow.	Output matches input.

Table 2. Grammar Rules for Form Follows Form (cont.)

	Tier 1		Tier 2	
Function	*Incoming Flow(s)*	*Outgoing Flow(s)*	*Incoming Flow(s)*	*Outgoing Flow(s)*
Process	Must be a status signal or control signal flow.	Output matches input.	Must be a status signal or control signal flow.	Output matches input.
Stabilize	Uses previous flow from function chain.	Output matches input.	Uses previous flow from function chain.	Output matches input.
Secure	Uses previous flow from function chain.	Output matches input.	Uses previous flow from function chain.	Output matches input.
Position	Uses previous flow from function chain.	Output matches input.	Uses previous flow from function chain.	Output matches input.

4 RESULTS: GENERATING A FUNCTIONAL MODEL

In this section two functional models using the Tier 1 and Tier 2 approaches are generated. Both functional models will utilize input components of an ice tea maker as shown in Section 3.1. Section 4.1 steps through the logic for creating a Tier 1 functional model and Section 4.2 presents the same information for a Tier 2 functional model. Finally, Section 4.3 compares machine generated functional models to a control model as well as discusses limitations of the 2-tier approach.

4.1 Generating a Tier 1 Functional Model

Tier 1 of FFF relies solely upon component chains entered by the user, repository data, and grammar rules presented in Section 3.3. Figure 3 shows the components identified in Figure 1 in a block representation. The component names are then replaced with their associated functionality and shown with their incoming and outgoing functionality (Table 1) in Figure 4. By stepping through the elements in each block in Figure 4 and combining the grammar rules from Table 2 a Tier 1 functional model can be constructed.

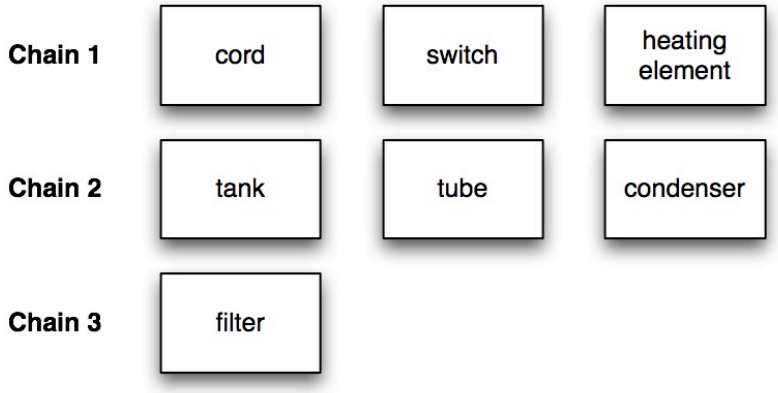


Figure 3. Component chains

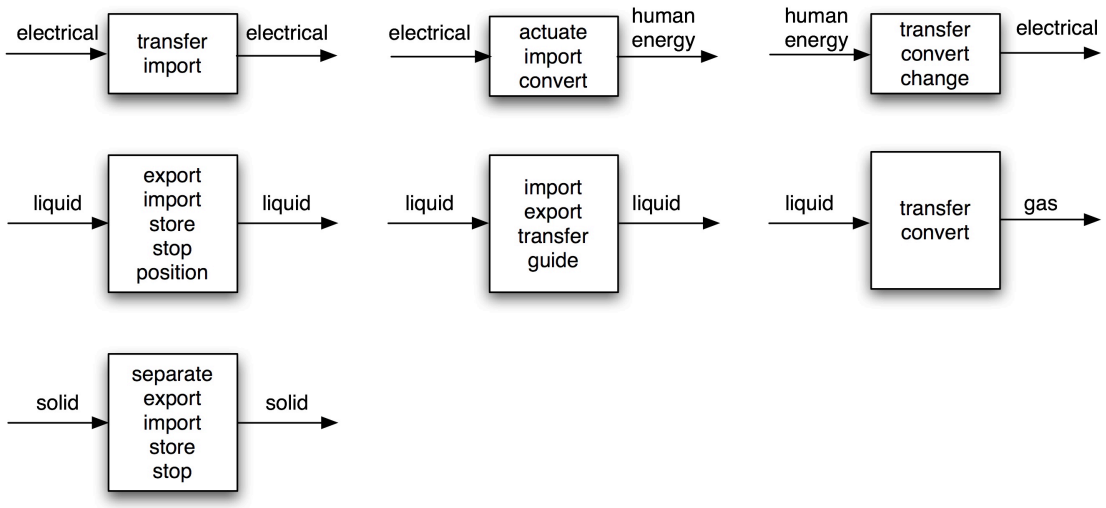


Figure 4. Functions and flows (Tier 1) associated with each component

Figure 5 shows the resulting functional model from the application of the grammar rules and the functions shown in Figure 4. Function chain 1 begins with the function import as specified by the import grammar rule. The flow of electrical energy is then attached because it is the highest ranked flow associated with the component cord (Table 2). Next, the function transfer is placed because it is the highest ranked function associated with a cord. Note that the function import is not again inserted in

the function chain because it can only exist at the beginning of a function chain. The component switch causes the addition of the functions actuate, convert, and transfers. Following the grammar rules a control signal is added as an additional input to the actuate function. The convert function continues to use electrical energy as the input flow and then chooses the human energy as an output flow. Human energy is the highest ranked flow for the component switch that does not match the input flow. The heating element component then adds the functions of transfer, convert, and change to the function chain. Finally, the function export is added as the last function in the component chain.

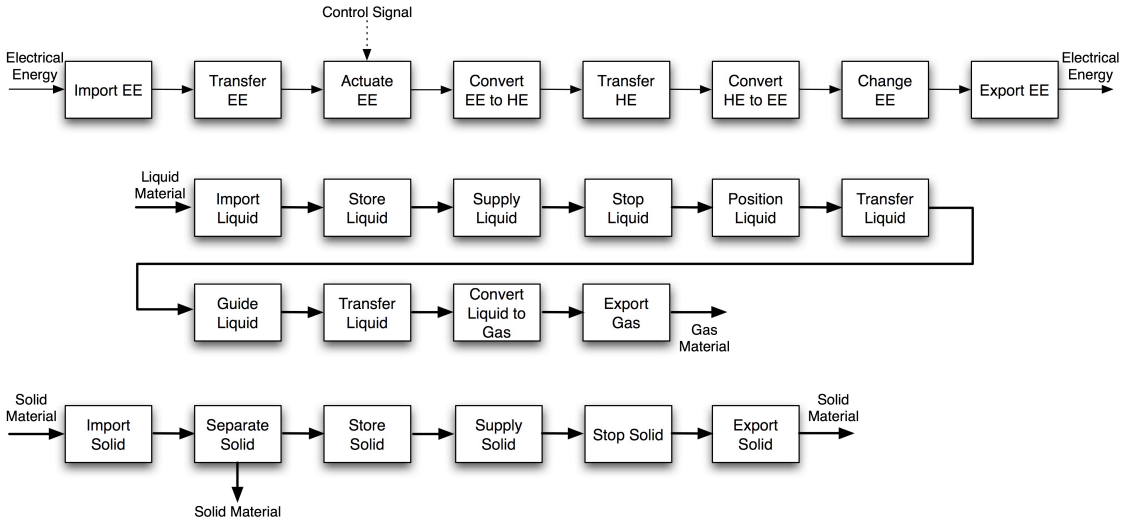


Figure 5. Tier 1 functional model of an ice tea maker

The second function chain is constructed in the same manner as the first function chain. An additional grammar rule is utilized directly following the store function. As listed in Table 2 the function supply must directly follow the function store, thus it is then inserted in the functional model. The third and final function chain makes use of another unique grammar rule. An additional duplicate output flow of solid material is

automatically added to the separate solid function. Conceptually this rule is obvious, if something is separate one would end up with a minimum of 2 separate or distinct items.

4.2 Generating a Tier 2 Functional Model

Tier 2 of FFF builds on the Tier 1 algorithm by allowing the user to specify the flows of a product. Again, the staring components emerge from Figure 3 and are augmented with the Tier 2 functions and flows in Figure 6. The resulting Tier 2 functional model is then shown in Figure 7. .

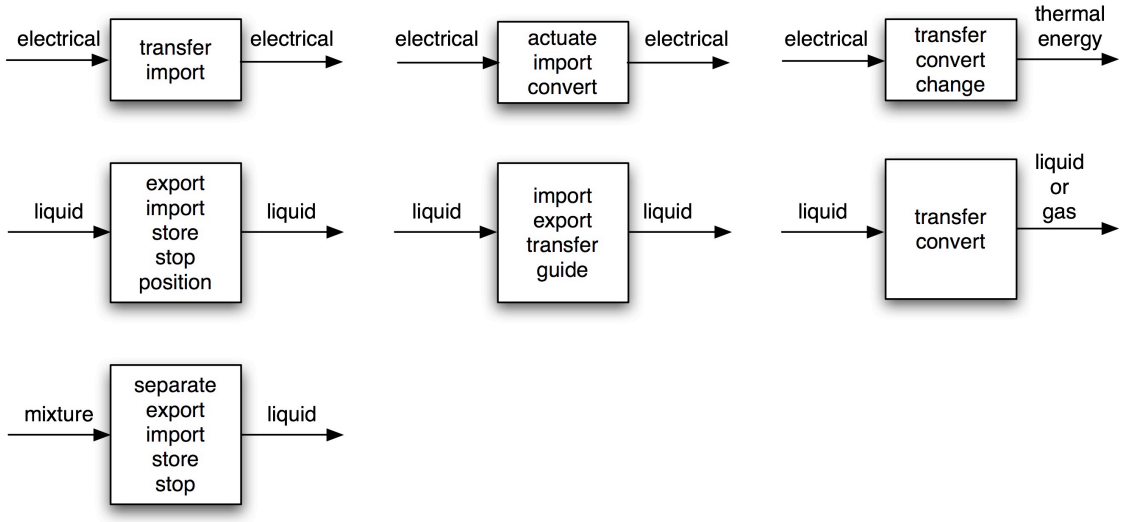


Figure 6. Functions and flows (Tier 2) associated with each component

The first chain in Figure 7 starts off much the same way as the Tier 1 functional model, however the user is allowed to specify the imported flow of electrical energy. The chain continues to mimic the Tier 1 chain until the convert function appears. It is important to note that in this function chain only a single convert function is present. The convert function associated with the switch has been removed. Tier 2 grammar rules for convert specify that the user is allowed to specify the output flow. In the case of the switch the output flow was selected to be the same as the input flow of electrical

energy, meaning that the user wanted electrical energy to continue in the function chain. Adhering to the grammar rules the function convert was then removed because the input flow matched the output flow. Another significant difference again occurs for the function convert associated with the heating element component. For this function the input flow is electrical energy, not human energy. Again, the user is allowed to specify the output flow and naturally chooses thermal energy (TE).

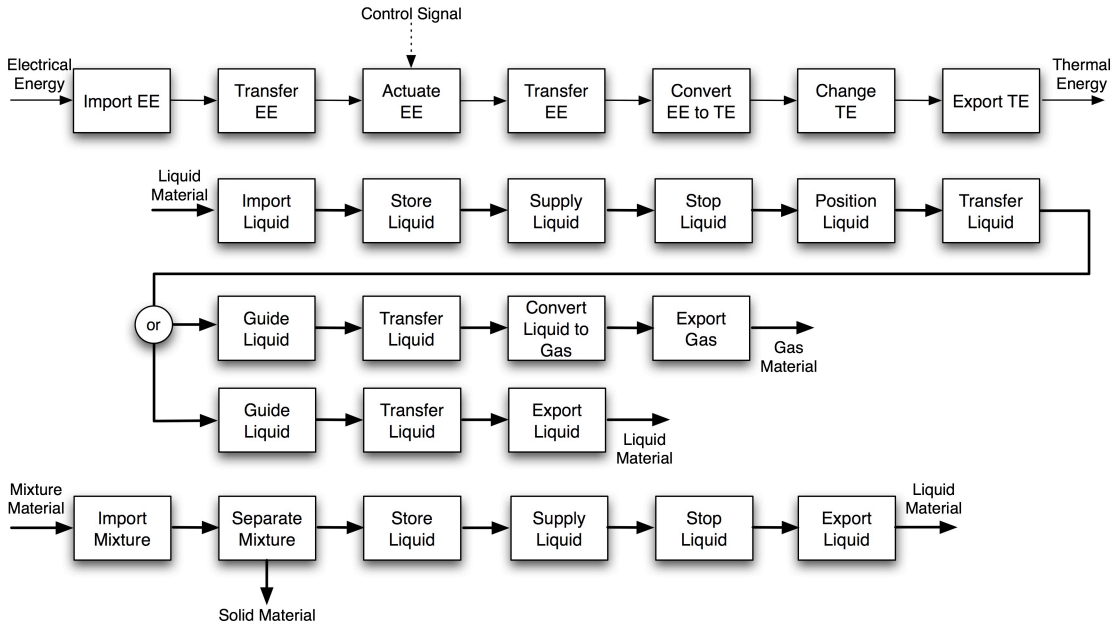


Figure 7. Tier 2 functional model of an ice tea maker

The second function chain is identical to the Tier 1 chain until the function convert realized by the condenser component. At this point the user is presented with a somewhat difficult choice on what to choose as a continuing output flow. The input flow is designated as a liquid component, but the purpose of a condenser is to convert a gas to a liquid. The reason a gas flow is not present in this function chain is because no component that converts a liquid to a gas is listed. If the user chooses the output flow of

gas the function convert will remain in the chain. However, selecting liquid as an output flow would then cause the convert function to be automatically removed. As a designer neither choice is particularly ideal, thus both flow paths are shown for this decision.

The third and final function chain mirrors the functionality of its Tier 1 counterpart, but begins by importing a mixture flow. The component filter is considered to be a basic coffee filter that would import both liquid and tea leaves. For the Tier 2 implementation it is expected that the user would then select mixture as an input flow. Another difference is shown by the separate function. Following the Tier 2 grammar set the user is allowed to choose the terminating output flow as well as the continuing output flow. Since the tea leaves will be discarding a solid material is shown as the terminating flow and a liquid shown as the continuing flow.

4.3 Discussion: Comparing FFF to Control Functional Models

Without much comparison it is most obvious that the Tier 1 and Tier 2 models contain several more functions than the control model presented in Figure 2. The purpose of the model in Section 3 was to demonstrate high-level functionality in order to show their associated components. Since FFF input was derived from Figure 2 we present a functional comparison of FFF to the original functional model.

Table 3 contains a listing of all of the functions present in the initial simplified functional model (Figure 2). In total the control model contains 15 unique functions and are shown in alphabetical order. An ‘x’ is placed in the table when functions between the FFF method and hand generated model match. In total 53.3% of the function-flow pairs are also realized by the Tier 1 algorithm and 80% appearing in the Tier 2 algorithm.

The Tier 1 algorithm fails to capture a significant portion of the functionality because of improper flows. For example the Tier 1 grammar allows for human energy to be converted to electrical energy instead of the more appropriate conversion of electrical energy to thermal energy. The Tier 1 algorithm also fails to import a mixture material, but instead imports a solid material. Results are substantially improved by using the

Tier 2 grammar, which allows the user to more accurately identify flows throughout the components.

Both tiers assume that only a single flow is passing between each function. That is connection information between discrete component chains is not captured. In order to accurately trace multiple and branching flows it will first be necessary to gather more information from the user regarding how components will interact with one another and currently remains as future work.

Table 3. Control and computer generated model comparison

Control	Tier 1	Tier 2
Actuate Electrical Energy	x	x
Convert Electrical Energy to Thermal Energy		x
Convert Gas to Liquid		
Convert Liquid to Gas	x	x
Export Liquid		x
Import Electrical Energy	x	x
Import Liquid	x	x
Import Mixture		x
Separate Mixture		x
Store Liquid	x	x
Supply Liquid	x	x
Transfer Electrical Energy	x	x
Transfer Gas		
Transfer Liquid	x	x
Transfer Thermal Energy		
	53.30%	80%

Output in the form a function adjacency matrix would then be used to seed one or both of the existing automated concept generators. It is expected that output from the Tier 1 approach may generate novel or unique concept variants not previously envisioned by the users. Their novelty would mostly be attributed to the automated

flow selection. The Tier 2 approach, however, would perhaps return more appropriate concepts. That is that their functionality and associated flows more closely match the user's initial intent.

5 CONCLUSIONS AND FUTURE WORK

From an engineering education standpoint, *design* is the perhaps one of the toughest topics to teach and, often, the most feared course assignment in a given engineering department. That is likely due to the emphasis of modern design techniques on abstracting the problem and identifying this fuzzy, hard to grasp (and explain) concept of functionality. Yet, this is the area of engineering where innovation takes root and where students need the most nurturing. Numerous studies have shown that early design is the best place in a product's life cycle to promote innovation, reduce risks, control costs and avoid delays. From research to education to practice, this form-initiated approach to concept generation will have a significant impact on engineering design innovation by offering a new paradigm for AI-based concept generation. Specifically, Form Follows Form will accept natural language input to seed an initial search for design alternatives and leverage the designer preferences inherent in that natural language to return tailored, innovative design concepts.

As shown in Section 4, FFF comes close to replicating a control functional model, but does not yet capture connections between discrete function chains. Future work includes the development of a Tier 3 Approach. This tier is analogous with a very detailed or directed search: I would like to see only journal papers published by a certain author in specific journals from May 1972 to July 1978. Tier 3 mimics the same approach of Tier 2 but also allows for designers to designate connections between components within separate component chains. An additional method to better realize user intent is to ask designers to specify the "commonness" of returned solutions. The underlying idea is that there is a spectrum of components that may solve a given function and, based upon observations in the repository, the algorithm can favor components that

solve a function most of the time (i.e., a common solution) or components that rarely solve a function (i.e., a uncommon solution). The uncommon solutions may in fact spur more innovative designs. There are two approaches to achieve this outcome. The first will correlate "commonness" with functionality. Uncommon functions would be functions that are rarely realized throughout the repository and overall are unique to a specific component. Common functions would be functions that solve a component a large percentage of the time, like the 70/30 allocation noted in Section 3. The second approach to "commonness" is to pass the desired level to the concept generation routines that will later be employed to generate new design alternatives. Both approaches will be explored for Form Follows Form. Use of FFF can also lead to ways to record product knowledge in a repository and also be used as a tool to teach functionality to engineering students.

ACKNOWLEDGEMENTS

This work is supported by the National Science Foundation under Grant No. CMMI-074267. Any opinions or findings of this work are those of the authors and do not necessarily reflect the view of the National Science Foundation or our collaborators.

6 REFERENCES

[1] Antonsson, E. and Cagan, J., 2001, *Formal Engineering Design Synthesis*, New York, NY, Cambridge University Press.

[2] Pahl, G. and Wallace, K., 2002, *Using the Concept of Functions to Help Synthesize Solutions*, London, Springer.

[3] Otto, K., 2001, "A process for modularizing product families," *International conference on engineering design, ICED 01*, Glasgow, Scotland.

[4] Shah, J.J., Vargas-Hernández, N., Summers, J.S., and Kulkarni, S., 2001, "Collaborative Sketching (C-Sketch) – An Idea Generation Technique for Engineering Design," *Journal of Creative Behavior*, **35**(3): 168-198.

[5] Roth, K., 2002, *Design Catalogs and Their Usage*, London, Springer.

[6] Ward, A., 1989, "A Theory of Quantitative Inference Applied to a Mechanical Design Compiler," *Doctoral Thesis,* Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA.

[7] Ward, A. and Seering, W., 1993, "Quantitative Inference in a Mechanical Design 'Compiler'," *Journal of Mechanical Design*, **115**: 29-35.

[8] McAdams, D.A. and Wood, K.L., 2002, "A Quantitative Similarity Metric for Design-by-Analogy," *Journal of Mechanical Design*, **124**(2): 173-182.

[9] Altshuller, G., 1984, *Creativity as an Exact Science*, Luxembourg, Gorden and Breach.

[10] Zwicky, F., 1969, *Discovery, Invention, Research - Through the Morphological Approach*, Toronto, The Macmillian Company.

[11] Pahl, G. and Beitz, W., 1996, *Engineering Design: A Systematic Approach*, Springer Verlag.

[12] Otto, K. and Wood, K., 2001, *Product Design: Techniques in Reverse Engineering, Systematic Design, and New Product Development*, New York, Prentice-Hall.

[13] Ullman, D.G., 2002, *The Mechanical Design Process 3rd Edition*, New York, McGraw-Hill, Inc.

[14] Ulrich, K.T. and Eppinger, S.D., 2004, *Product Design and Development*, Boston, MA, McGraw-Hill/Irwin.

[15] Cutherell, D., 1996, "Chapter 16: Product Architecture," in *The PDMA Handbook of New Product Development*, M. Rosenau Jr., Editor, Wiley and Sons.

[16] Hubka, V. and Ernst Eder, W., 1984, *Theory of Technical Systems*, Berlin, Springer-Verlag.

[17] Otto, K. and Wood, K., 1996, "A Reverse Engineering and Redesign Methodology for Product Evolution," *Proceedings of the 1996 ASME Design Theory and Methodology Conference*, **96-DETC/DTM-1523**, Irvine, CA.

[18] Otto, K. and Wood, K., 1997, "Conceptual and Configuration Design of Products and Assemblies," in *ASM Handbook, Materials Selection and Design*, ASM International.

[19] Pimmler, T. and Eppinger, S. (1994). *Integration Analysis of Product Decompositions*. ASME Design Engineering Technical Conferences & Computers and Information in Engineering Conference.

[20] Schmidt, L. and Cagan, J., 1995, "Recursive Annealing: A Computational Model for Machine Design," *Research in Engineering Design*, **7**(2): 102-125.

[21] Shimomura, Y., Tanigawa, S., Takeda, H., Umeda, Y., and Tomiyama, T., 1996, "Functional Evaluation Based on Function Content," *Proceedings of the 1996 ASME Design Theory and Methodology Conference*, **96-DETC/DTM-1532**, Irvine, CA.

[22] Radcliffe, D. and Lee, T.Y., 1989, "Design Methods Used by Undergraduate Engineering Students," *Design Studies*, **10**(4): 199-207.

[23] Cross, N., 1994, *Engineering design methods: Strategies for product design, 2nd Ed.*, Chichester, UK, John Wiley & Sons.

[24] Ivashkov, M., 2004, "ACCEL: a Tool Supporting Concept Generation in the Early Design Phase," *PhD Thesis*, The Eindhoven University of Technology, Eindhoven, The Netherlands.

[25] Wodehouse, A., Grierson, H., Ion, W.J., Juster, N., Lynn, A., and Stone, A.L., 2004, "Tikiwiki: A Tool to Support Engineering Design Students in Concept Generation," *International Engineering and Product Design Education Conference*, Delft, Netherlands.

[26] Sridharan, P. and Campbell, M., 2005, "A Study on the Grammatical Construction of Function Structures," *Artificial Intelligence in Engineering Design, Analysis and Manufacture*, **19**(3): 139-160.

[27] Du, X. and Chen, W., 2004, "Sequential Optimization and Reliability Assessment for Probabilistic Design," *Journal of Mechanical Design*, **126**: 225-233.

[28] 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): 65-82.

[29] Stone, R. and Wood, K., 2000, "Development of a Functional Basis for Design," *Journal of Mechanical Design*, **122**(4): 359-370.

[30] Bryant, C., McAdams, D., Stone, R., Kurtoglu, T., and Campbell, M., 2005, "A Computational Technique for Concept Generation," *Proceedings of IDETC/CIE 2005*, **DETC2005-85323**, Long Beach, CA.

[31] Bryant, C., Stone, R., McAdams, D., Kurtoglu, T., and Campbell, M., 2005, "Concept Generation from the Functional Basis of Design," *International Conference on Engineering Design, ICED 05*, Melbourne, Australia.

[32] Kurtoglu, T. and Campbell, M., 2008, "Automated Synthesis of Electromechanical Design Configurations from Empirical Analysis of Function to Form Mapping," *Journal of Engineering Design*, **20**(1): 83-104.

[33] Kurtoglu, T., Campbell, M., Gonzalez, J., Bryant, C., Stone, R., and McAdams, D., 2005, "Capturing Empirically Derived Design Knowledge for Creating Conceptual Design Configurations," *Proceedings of IDETC/CIE 2005*, **DETC2005-84405**, Long Beach, CA.

[34] Kurtoglu, T., Campbell, M., Bryant, C., Stone, R., and McAdams, D., 2008, "A Component Taxonomy as a Framework for Computational Design Synthesis," *Journal of Computers and Information Science in Engineering*, **Accepted, to appear**.

[35] Kurtoglu, T., Campbell, M., Bryant, C., Stone, R., and McAdams, D., 2005, "Deriving a Component Basis for Computational Functional Synthesis," *International Conference on Engineering Design, ICED'05*, Melbourne, Australia.

[36] Bryant, C., McAdams, D., Stone, R., Kurtoglu, T., and Campbell, M., 2006, "A Validation Study of an Automated Concept Generator Design Tool," *Proceedings of IDETC/CIE 2006*, **DETC2006-99489**, Philadelphia, PA.

[37] Bryant, C., Pieper, E., Walther, B., Kurtoglu, T., Stone, R., McAdams, D., and Campbell, M., 2006, "Software Evaluation of an Automated Concept Generator Design Tool," *Proceedings of the 2006 ASEE Annual Conference*, **ASEE-2006-1758**, Chicago, IL.

[38] Kurtoglu, T. and Campbell, M. (2007). *Exploring the worth of automatically generated design alternatives based on designer preferences*. International Conference on Engineering Design, Paris, France.

[39] Kurtoglu, T. and Campbell, M., 2008, "An Evaluation Scheme for Assessing the Worth of Automatically Generated Design Alternatives," *Research in Engineering Design*, **Accepted, To Appear**.

[40] Bohm, M., Stone, R., and Szykman, S., 2005, "Enhancing Virtual Product Representations for Advanced Design Repository Systems," *Journal of Computer and Information Science in Engineering*, **5**(4): 360-372.

[41] Szykman, S., Sriram, R., and Regli, W., 2001, "The Role of Knowledge in Next-generation Product Development Systems," *Journal of Computer and Information Science in Engineering*, **1**(1): 3-11.

[42] Szykman, S., Fenves, S., Keirouz, W., and Shooter, S., 2001, "A Foundation for Interoperability in Next-Generation Product Development Systems," *Computer-Aided Design*, **33**(7): 545-559.

[43] Bohm, M., Stone, R., Simpson, T., and Steva, E., 2008, "Introduction of a Data Schema: To Support a Design Repository," *Computer-Aided Design*, **40**(7): 801-811.

[44] Shooter, S., Simpson, T., Kumara, S., Stone, R., and Terpenny, J., 2005, "Toward a Multi-Agent Information Management Infrastructure for Product Family Planning and Mass Customisation," *International Journal for Mass Customisation*, **1**(1): 134-155.

[45] Bohm, M., Vuchovich, J., and Stone, R., 2007, "An Open Source Application for Archiving Product Design Information," *Proceedings of DETC'07*, **DETC2007-35401**, Las Vegas, NV.

[46] Bohm, M. and Stone, R., 2009, "A Natural Language to Component Term Methodology: Towards a Form Based Concept Generation Tool," *ASME Design Engineering and Technical Conference*, San Diego, CA.

[47] Nagel, R., Vucovich, J., Stone, R., and McAdams, D., 2008, "A Signal Grammar to Guide Functional Modeling of Electromechanical Products," *Journal of Mechanical Design*, **130**(5): 051101-1-10.

[48] Russell, S.J. and Norvig, P., 2003, *Artificial Intelligence: A Modern Approach (2nd ed.)*, Upper Saddle River, NJ, Prentice Hall.

[49] Poole, D., Mackworth, A., and Goebel, R., 1998, *Computational Intelligence: A Logical Approach*, New York, Oxford University Press.

[50] Luger, G. and Stubblefield, W., 2004, *Artificial Intelligence: Structures and Strategies for Complex Problem Solving (5th ed.)*, The Benjamin/Cummings Publishing Company, Inc.

[51] Sen, C., Caldwell, B., Summers, J., and Mocko, G., 2009, "Evaluation of the Functional Basis using an Information Theoretic Approach," *Submitted to Journal of Artifical Intelligence for Engineering Design, Analysis and Manufacturing*.

2. CONCLUSIONS

The philosophical contribution of this dissertation research is to transform the process of engineering design. The transformation occurs through formulating algorithms and harnessing computational resources heretofore relegated to the fields of computer science and statistics. Specifically, the key contributions of this body of research are the:

- Formulation and implementation of a design repository schema that is robust and able to capture current product design knowledge for archival and retrieval purposes;
- Ability to capture design information in the repository organized into seven main categories that include: artifact-, function-, failure-, physical-, performance-, sensory- and media-related information types;
- Design of a database schema for easy expansion to capture additional types of design information;
- Design and development of a novel data entry application to support product design knowledge archival, reuse and product dissection activities;
- Cross-platform entry application, known as the Repository Entry Application, released under the GNU public license and, thus, available for modification by other researchers;
- Adoption of the Repository Entry Application at over ten top-tier research institutions in the United States;
- Formulation of the first computationally driven function-based concept generation application that allows mechanical design to be performed more thoroughly and quickly than by manual methods;
- First evaluation of the computationally driven concept generation solutions for their impact on designer creativity;
- Finding that solutions suggested by the computational approach capture an overwhelming majority of human generated solutions while relying upon a relatively small knowledge set of product information;

- Formulation of an artificial intelligence based approach to translate natural language component descriptions into computer parse-able terms;
- Generation of an initial set of natural language to component naming terms that conforms to a Pareto frontier;
- Formulation of a novel form-initiated approach to concept generation called Form Follows Form;
- Artificial intelligence based approach in Form Follows Form that transforms an initial component solution seed for a design problem to its underlying functionality to broaden the search for alternative solutions; and
- Completely automated transformation of form-to-function-to-form and capture of designer preference for a more directed solution approach by Form Follows Form.

Collectively, the papers included in this dissertation describes how the burdens often associated with conceptual phase of the engineering design process may be overcome with intelligent algorithms and computational power. As shown in many studies it is imperative for engineers and designers to have access to as much information as possible during the conceptual phases of design. Proper use of such information results in quicker and less costly development times as well as a quality product or process. As a whole, engineering design methods have been developed to assist in nearly every phase of the design process from customer needs analysis to design for lifecycle and reliability. The goal of this work is to bring to bear the power of computational thinking on the early phases of the design process. The work presented in the included papers begins by outlining a method to capture and store product design information in the form of a design repository and concludes with methods to aid in conceptual development and preliminary design analysis.

The repository project at Missouri S&T has significantly impacted and contributed to engineering design knowledge as well as the broader field of science. Directly, the repository has transformed a disparate set of product design knowledge into a coherent body. By using the repository system and data contained within, new

methods for concept generation can be explored. The implementation of the design repository on the web, supports a new mode of design knowledge exchange between researchers in both industry and academia. The exchange of ideas and information furthers the development of the repository project by incorporating supplemental design knowledge components. Each contribution heightens the resolution as well as the breadth of design knowledge within the repository. In addition, recent independent studies by external researches have assessed the information content of the repository and found the repository data to be both useful to designers and impact the design process in a positive way [52, 53]

Two such methods enabled by the design repository, explored within this work, are the morphological search tool and the Form Follows Form method. Without the design repository and wealth of engineering design information contained within neither of these methods would be possible. The morphological search tool was the first computational concept generation tool built to take advantage of the design repository. For the first time users were able to quickly specify desired functionality and be quickly presented with a set of possible solution components. Other researchers have since enhance the morphological search tool by incorporating component connection information.

Form Follows Form is the next enhancement to conceptual level design tools. More specifically, FFF allows access to a host of engineering design methodologies such as preliminary failure mode analysis, preliminary risk analysis, two discrete automated concept generation methods, as well as general function identification. FFF enables novice and expert designers to specify components using natural language removing the need to be well versed in specific function or component taxonomies. In addition FFF establishes a framework for artificial intelligence and reasoning in design. The logic and grammar rules in FFF serve as a foundation for automated reasoning in design.

Together, broader impacts of this research include the underlying design knowledge segmentation and categorization techniques as well as building a foundation to support automated reasoning. The underpinnings of the repository increase the ability to archive any corporate knowledge of human experts in a form that is parsable and computable. The form-initiated approach to concept generation will have a significant impact on engineering design innovation by offering a new paradigm for AI-based concept generation.

The five publications constituting this body of work do, jointly, prove the original hypothesis of this research:

> *Computational thinking (i.e., product design knowledge archival and reuse and AI algorithms) can be applied in the early phases of design to increase the quantity, quality, and breadth of concept variants produced during the design of a product.*

More generally Table 2.1 lists the relevant research works by the author on the subjects of transforming engineering design through artificial intelligence based algorithms and computational thinking.

Table 2.1. List of publications

1. Bohm, M., Stone, R., 2009, "A Natural Language to Component Term Methodology: Towards a Form Based Concept Generation Approach," *Submitted to Proceedings of IDETC/CIE 2009,* DETC2009/CIE-86581, San Diego, CA.
2. Bohm, M., Stone, R., 2009 "Form Follows Form – Is a New Paradigm Needed?," *Submitted to Proceedings of the IMECE '09*, IMECE2009-10410, Lake Buena Vista, FL.
3. Bohm, M., Stone, R., Simpson, S. and Steva, L., 2008 "Introduction of a Data Schema: The Inner Workings of a Design Repository," *Journal of Computer Aided Design.*, In press, doi:10.1016/j.cad. 2008.09.003
4. Bohm, M., Vucovich, J. and Stone, R., 2008, " Using a Design Repository to Drive Concept Generation," *Journal of Computer and Information Science in Engineering*, **8**(1):14502.

Table 2.1. List of publications

5. Stroble, J., Nagel, R., Poppa, K., Bohm, R. and Stone, R., 2008, "A Retrospective on Twenty Years of the Design Theory and Methodology Conference," *Proceedings of IDETC/CIE 2008*, DETC2008/DTM-49373, Brooklyn, NY.
6. Nanda, J., Henri, J., Simpson, T., Stone, R., Bohm, M. and Shooter, S., 2007, "Product Family Design Knowledge Representation, Aggregation, Reuse, and Analysis," *Artificial Intelligence in Engineering Design, Analysis and Manufacture*, **21**(2):173-192.
7. Bohm, M., Vucovich, J. and Stone, R., 2007 "An Open Source Application for Archiving Product Design Information," *Proceedings of IDETC/CIE 2007,* DETC2007-35401, Las Vegas, NV.
8. Bryant, C., McAdams, D., Bohm, M. and Stone, R., 2007 "An Interactive Computational Design Tool: A Hybrid of Two Methods," *Proceedings of IDETC/CIE 2007,* DETC2007-35583, Las Vegas, NV.
9. Nagel, R., Bohm, M., Stone, R. and McAdams, D., 2007 "A Representation of Carrier Flows for Functional Design," *Proceedings of the International Conference on Engineering Design,* ICED 07, Paper 635, Paris, France.
10. Bohm, M., Stone, R., Simpson, S. and Steva, L., 2006 "Introduction of a Data Schema: The Inner Workings of a Design Repository," *Proceedings of IDETC/CIE 2006*, DETC2006-99518, Philadelphia, PA.
11. Bohm, M., Stone, R. and Szykman, S., 2005, "Enhancing Virtual Product Representations for Advanced Design Repository Systems," *Journal of Computer and Information Science in Engineering*, **5**(4):360-372.
12. Bohm, M., Vucovich, J., and Stone, R., 2005, "Capturing Creativity: Using a Design Repository to Drive Concept Innovation," *Proceedings of IDETC/CIE 2005*, DETC2005-85105, Long Beach, CA.
13. Van Wie, M., Bryant, C., Bohm, M., McAdams, D. and Stone, R., 2004, "A general model of function-based representations," *Artificial Intelligence in Engineering Design, Analysis and Manufacture*, **19**(2):89-111.
14. Bohm, M., and Stone, R., 2004, "Representing Functionality to Support Reuse: Conceptual and Supporting Functions," *Proceedings of DETC'04*, DETC2004-57693, Salt Lake City, UT.
15. Bohm, M. and Stone, R., 2004, "Product Design Support: Exploring a Design Repository System," *Proceedings of IMECE'04*, IMECE2004-61746, Anaheim, CA.
16. Stock, M., Bohm, M., Stone, R. and Hubing, N., 2003, "Using Product Architecture-Based Design Methods to Get Smart in the Battlefield," *Proceedings of the International Conference on Engineering Design*, ICED 03, Paper 1192, Stockholm, Sweden.
17. Bohm, M., Stone, R. and Szykman, S., 2003, "Enhancing Virtual Product Representations for Advanced Design Repository Systems," *Proceedings of DETC2003*, DETC2003/CIE-48239, Chicago, IL.

From an engineering standpoint, *design* is the perhaps one of the toughest topics to truly grasp. This difficulty is likely due to the emphasis of modern design techniques

on abstracting the problem and identifying this fuzzy, hard to explain concept of functionality. Yet, this is the area of engineering where innovation takes root and where minds need the most nurturing. Numerous studies have shown that early design is the best place in a product's life cycle to promote innovation, reduce risks, control costs and avoid delays.

In summary, the design repository, morph matrix search, and Form Follows Form methodologies will have a significant impact on engineering design innovation by offering a new paradigm for AI-based concept generation. As a result of this new application of computational thinking in a previously manual phase of engineering design, this research has democratized the concept generation process and increased the number, quality and breadth of concept variants that can be generated by any engineering designer. Finally, third party validation of the work is in progress and is already finding utility in the methods.

3. REFERENCES

[1] Szykman, S., Sriram, R., and Smith, S. (1996), Proceedings of the NIST Design Repository Workshop, Gaithersburg, MD, National Institute of Standards and Technology, November.

[2] Murdock, J., Szykman, S., and Sriram, R., 1997, "An Information Modeling Framework to Support Design Databases and Repositories," *Proceedings of DETC'97*, **DETC97/DFM-4373**, Sacramento, CA.

[3] Szykman, S., Racz, J., and Sriram, R., 1999, "The Representation of Function in Computer-Based Design," *Proceedings of the ASME Design Theory and Methodology Conference*, **DETC99/DTM-8742**, Las Vegas, NV.

[4] Shooter, S., Keirouz, W., Szykman, S., and Fenves, S., 2000, "A Model For Information Flow In Design," *Proceedings of the ASME Design Theory and Methodology Conference*, **DETC2000/DTM-14550**, Baltimore, MD.

[5] Szykman, S., 2002, "Architecture and Implementation of a Design Repository System," *Proceedings of DETC2002*, **DETC2002/CIE-34463**, Montreal, Canada.

[6] Svensson, D. and Malmqvist, J., 2001, "Integration of Requirement Management and Product Data Management Systems," *Proceedings of DETC2001*, **DETC2001/CIE-21246**, Pittsburgh, PA.

[7] Bohm, M., Stone, R., and Szykman, S., 2005, "Enhancing Virtual Product Representations for Advanced Design Repository Systems," *Journal of Computer and Information Science in Engineering*, **5**(4): 360-372.

[8] Szykman, S., Sriram, R., and Regli, W., 2001, "The Role of Knowledge in Next-generation Product Development Systems," *Journal of Computer and Information Science in Engineering*, **1**(1): 3-11.

[9] Szykman, S., Fenves, S., Keirouz, W., and Shooter, S., 2001, "A Foundation for Interoperability in Next-Generation Product Development Systems," *Computer-Aided Design*, **33**(7): 545-559.

[10] Bryant, C., McAdams, D., Stone, R., Kurtoglu, T., and Campbell, M., 2005, "A Computational Technique for Concept Generation," *Proceedings of IDETC/CIE 2005*, **DETC2005-85323**, Long Beach, CA.

[11] Bryant, C., Stone, R., McAdams, D., Kurtoglu, T., and Campbell, M., 2005, "Concept Generation from the Functional Basis of Design," *International Conference on Engineering Design, ICED 05*, Melbourne, Australia.

[12] Bohm, M., Stone, R., Simpson, T., and Steva, E., 2008, "Introduction of a Data Schema: To Support a Design Repository," *Computer-Aided Design*, **40**(7): 801-811.

[13] Shooter, S., Simpson, T., Kumara, S., Stone, R., and Terpenny, J., 2005, "Toward a Multi-Agent Information Management Infrastructure for Product Family Planning and Mass Customisation," *International Journal for Mass Customisation*, **1**(1): 134-155.

[14] Bohm, M., Vuchovich, J., and Stone, R., 2007, "An Open Source Application for Archiving Product Design Information," *Proceedings of DETC'07*, **DETC2007-35401**, Las Vegas, NV.

[15] Antonsson, E. and Cagan, J., 2001, *Formal Engineering Design Synthesis*, New York, NY, Cambridge University Press.

[16] Pahl, G. and Wallace, K., 2002, *Using the Concept of Functions to Help Synthesize Solutions*, London, Springer.

[17] Otto, K., 2001, "A process for modularizing product families," *International conference on engineering design, ICED 01*, Glasgow, Scotland.

[18] Shah, J.J., Vargas-Hernández, N., Summers, J.S., and Kulkarni, S., 2001, "Collaborative Sketching (C-Sketch) – An Idea Generation Technique for Engineering Design," *Journal of Creative Behavior*, **35**(3): 168-198.

[19] Roth, K., 2002, *Design Catalogs and Their Usage*, London, Springer.

[20] Ward, A., 1989, "A Theory of Quantitative Inference Applied to a Mechanical Design Compiler," *Doctoral Thesis,* Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA.

[21] Ward, A. and Seering, W., 1993, "Quantitative Inference in a Mechanical Design 'Compiler'," *Journal of Mechanical Design*, **115**: 29-35.

[22] McAdams, D.A. and Wood, K.L., 2002, "A Quantitative Similarity Metric for Design-by-Analogy," *Journal of Mechanical Design*, **124**(2): 173-182.

[23] Altshuller, G., 1984, *Creativity as an Exact Science*, Luxembourg, Gorden and Breach.

[24] Zwicky, F., 1969, *Discovery, Invention, Research - Through the Morphological Approach*, Toronto, The Macmillian Company.

[25] Pahl, G. and Beitz, W., 1996, *Engineering Design: A Systematic Approach*, Springer Verlag.

[26] Otto, K. and Wood, K., 2001, *Product Design: Techniques in Reverse Engineering, Systematic Design, and New Product Development*, New York, Prentice-Hall.

[27] Ullman, D.G., 2002, *The Mechanical Design Process 3rd Edition*, New York, McGraw-Hill, Inc.

[28] Ulrich, K.T. and Eppinger, S.D., 2004, *Product Design and Development*, Boston, MA, McGraw-Hill/Irwin.

[29] Cutherell, D., 1996, "Chapter 16: Product Architecture," in *The PDMA Handbook of New Product Development*, M. Rosenau Jr., Editor, Wiley and Sons.

[30] Hubka, V. and Ernst Eder, W., 1984, *Theory of Technical Systems*, Berlin, Springer-Verlag.

[31] Otto, K. and Wood, K., 1996, "A Reverse Engineering and Redesign Methodology for Product Evolution," *Proceedings of the 1996 ASME Design Theory and Methodology Conference*, **96-DETC/DTM-1523**, Irvine, CA.

[32] Otto, K. and Wood, K., 1997, "Conceptual and Configuration Design of Products and Assemblies," in *ASM Handbook, Materials Selection and Design*, ASM International.

[33] Pimmler, T. and Eppinger, S. (1994). *Integration Analysis of Product Decompositions*. ASME Design Engineering Technical Conferences & Computers and Information in Engineering Conference.

[34] Schmidt, L. and Cagan, J., 1995, "Recursive Annealing: A Computational Model for Machine Design," *Research in Engineering Design*, **7**(2): 102-125.

[35] Shimomura, Y., Tanigawa, S., Takeda, H., Umeda, Y., and Tomiyama, T., 1996, "Functional Evaluation Based on Function Content," *Proceedings of the 1996 ASME Design Theory and Methodology Conference*, **96-DETC/DTM-1532**, Irvine, CA.

[36] Radcliffe, D. and Lee, T.Y., 1989, "Design Methods Used by Undergraduate Engineering Students," *Design Studies*, **10**(4): 199-207.

[37] Cross, N., 1994, *Engineering design methods: Strategies for product design, 2nd Ed.*, Chichester, UK, John Wiley & Sons.

[38] Ivashkov, M., 2004, "ACCEL: a Tool Supporting Concept Generation in the Early Design Phase," *PhD Thesis*, The Eindhoven University of Technology, Eindhoven, The Netherlands.

[39] Wodehouse, A., Grierson, H., Ion, W.J., Juster, N., Lynn, A., and Stone, A.L., 2004, "Tikiwiki: A Tool to Support Engineering Design Students in Concept Generation," *International Engineering and Product Design Education Conference*, Delft, Netherlands.

[40] Sridharan, P. and Campbell, M., 2005, "A Study on the Grammatical Construction of Function Structures," *Artificial Intelligence in Engineering Design, Analysis and Manufacture*, **19**(3): 139-160.

[41] Du, X. and Chen, W., 2004, "Sequential Optimization and Reliability Assessment for Probabilistic Design," *Journal of Mechanical Design*, **126**: 225-233.

[42] 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): 65-82.

[43] Stone, R. and Wood, K., 2000, "Development of a Functional Basis for Design," *Journal of Mechanical Design*, **122**(4): 359-370.

[44] Kurtoglu, T. and Campbell, M., 2008, "Automated Synthesis of Electromechanical Design Configurations from Empirical Analysis of Function to Form Mapping," *Journal of Engineering Design*, **20**(1): 83-104.

[45] Kurtoglu, T., Campbell, M., Gonzalez, J., Bryant, C., Stone, R., and McAdams, D., 2005, "Capturing Empirically Derived Design Knowledge for Creating Conceptual Design Configurations," *Proceedings of IDETC/CIE 2005*, **DETC2005-84405**, Long Beach, CA.

[46] Kurtoglu, T., Campbell, M., Bryant, C., Stone, R., and McAdams, D., 2008, "A Component Taxonomy as a Framework for Computational Design Synthesis," *Journal of Computers and Information Science in Engineering*, **Accepted, to appear**.

[47] Kurtoglu, T., Campbell, M., Bryant, C., Stone, R., and McAdams, D., 2005, "Deriving a Component Basis for Computational Functional Synthesis," *International Conference on Engineering Design, ICED'05*, Melbourne, Australia.

[48] Bryant, C., McAdams, D., Stone, R., Kurtoglu, T., and Campbell, M., 2006, "A Validation Study of an Automated Concept Generator Design Tool," *Proceedings of IDETC/CIE 2006*, **DETC2006-99489**, Philadelphia, PA.

[49] Bryant, C., Pieper, E., Walther, B., Kurtoglu, T., Stone, R., McAdams, D., and Campbell, M., 2006, "Software Evaluation of an Automated Concept Generator Design Tool," *Proceedings of the 2006 ASEE Annual Conference*, **ASEE-2006-1758**, Chicago, IL.

[50] Kurtoglu, T. and Campbell, M. (2007). *Exploring the worth of automatically generated design alternatives based on designer preferences*. International Conference on Engineering Design, Paris, France.

[51] Kurtoglu, T. and Campbell, M., 2008, "An Evaluation Scheme for Assessing the Worth of Automatically Generated Design Alternatives," *Research in Engineering Design*, **Accepted, To Appear**.

[52] Sen, C., Caldwell, B., Summers, J., and Mocko, G., 2009, "Evaluation of the Functional Basis using an Information Theoretic Approach," *Submitted to Journal of Artifical Intelligence for Engineering Design, Analysis and Manufacturing*.

[53] Thomas, J., Sen, C., Mocko, G., Summers, J., and Fadel, G., 2009, "Investigation of the Interpretability of Three Function Structure Representations: A User Study," *Submitted to ASME Design Entineering and Technical Conference*, San Diego, CA.

VITA

Matt Robert Bohm was born on August 20th, 1979 in Joplin, Missouri. The son of Roger and Bobbie Bohm grew up in Carthage, Missouri and attended Carthage public schools from kindergarten until graduation. In high school, Matt excelled at physics and math and was a founding member of Carthage High School's solar car team. His interest in math and science led him to the University of Missouri – Rolla in the fall of 1997 in pursuit of a degree in Mechanical Engineering. After completing his bachelor's degree, Matt extended his tenure at UMR in pursuit of a master's degree in Mechanical Engineering completed in May, 2004. Matt continued his study and earned a Ph.D. in Mechanical Engineering from the Missouri University of Science in Technology in May, 2009.