

Enhancing Virtual Product Representations for Advanced Design Repository Systems

Matt R. Bohm

Robert B. Stone

Ph.D.

Design Engineering Laboratory, Department of
Basic Engineering, University of Missouri—
Rolla, Rolla, Missouri 65401-0210

Simon Szykman

Ph.D.

Department of Homeland Security, Science and
Technology Directorate, Washington, DC 20528

This paper describes the transformation of an existing set of heterogeneous product knowledge into a coherent design repository that supports product design knowledge archival and web-based search, display, and design model and tool generation. Guided by design theory, existing product information was analyzed and compared against desired outputs to ascertain what information management structure was needed to produce design resources pertinent to the design process. Several test products were catalogued to determine what information was essential without being redundant in representation. This set allowed for the creation of a novel single point of entry application for product information and the development of a relational database for design knowledge archival. Web services were then implemented to support design knowledge retrieval through search, browse, and real-time design tool generation. Further explored in this paper are the fundamental enabling technologies of the design repository system. Additionally, repository-generated design tools are scrutinized alongside human-generated design tools for validation. Through this process researchers have been able to improve the way in which artifact data are gathered, archived, distributed and used. [DOI: 10.1115/1.1884618]

1 Introduction

As products become more complex there is an increased need for the designer or team of designers to have access to a breadth of design information spanning a variety of disciplines. Consideration of many types of artifacts is necessary when searching for component solutions, in order to ensure a high-quality product that meets the needs of the customer. Well suited to meet this need are design repositories—knowledge bases of heterogeneous product design knowledge that can be searched and reused. Note, we define the term “knowledge base” as the collection of design information elements that is used to populate the repository which is implemented in a database. Design knowledge refers to ideas inferred from information and represents an understanding of what is known about design.

Over the course of several years of research and integrated design coursework at UMR, a body of product design knowledge was developed for approximately 50 consumer products. This knowledge base, which included descriptive product information such as functionality, bills of materials, and design structure matrices, lacked a standard interface, data consistency, and the ability to output design models and tools with ease. Observing user interactions with the knowledge base, including design modeling activities as well as retrieval and reuse/redesign activities, revealed that these drawbacks served as a barrier to effective use of the knowledge base. By unifying these disparate components into a design repository, it has been possible to improve the utility of the knowledge base for viewing, searching, and reusing the wealth of preexisting design knowledge.

This paper reports on research efforts conducted to (1) identify the models of design knowledge required to support contemporary designer activities and the associated tools employed; (2) represent information from this product knowledge base in a design repository system; and (3) provide an architecture to enable design knowledge reuse in future product design. Specific goals of

this project were to create a system that could output design aids such as detailed bills of materials and matrices to support design computations. We refer to these elements as *design tools*. A single, simplified point of entry for product information that integrates well with product dissection processes [1] was also desired. Note that the product information to be archived is best termed as *design models* of the product. Collectively, design models and tools comprise the *design knowledge* contained within the repository. With these objectives identified, this paper examines the archive of design knowledge previously gathered and looks at how product information is collected and what design information is necessary to support design tool generation. Two key design models for archival purposes are functional models and a detailed bill of materials. These two sets of data are the foundation for transforming an emerging design information archive to a more mature design repository-based tool.

The paper begins by examining design aids and representations commonly used in modern design methods. To better understand the current space of design knowledge representation, a review of external design repository schemas, Product Data Management (PDM) systems, and commercially available software is conducted. Insight derived from knowledge base systems, design theory, and in-house product dissection experience are used to drive the development of the design repository system. The software development review includes analysis of commercial and prototype database and web options. With software component features summarized, customer needs are used to evaluate and select software technologies for the design repository. The underlying data structure and relationships of the design repository system are also reviewed with an emphasis on data consistency. An operational repository system is then demonstrated including data entry and representation conventions, database storage, and web-based features. Through the web-based repository browse and search, product information availability is revealed. Comparing human-generated design tools to their respective repository generated analogs verifies the design tool generation algorithms.

2 Background

In order to create a quality design repository, there are several elements of previous research that must be used to consistently

Contributed by the Committee for publication in the Journal of Computing and Information in Science Engineering. Manuscript received July 21, 2003; Revised February 9, 2005. Associate Editor: R. Crawford.

represent and efficiently store product information. This section reviews external design repository implementations, Product Data Management (PDM) systems, and commercially available software applications that resemble design repository systems. Next, Sec. 2.2 focuses on product functionality, a key component of design information for product categorization, search, and reuse. In particular, the functional basis is described as a means to represent product functionality. Finally, Sec. 2.3 outlines commonly used function-based design tools. The specific design tools mentioned include bills of materials (BOM), design structure matrices (DSM), function component matrices (FCM), and functional models.

2.1 Similar Systems. Several researchers, institutions, and corporations have developed systems to capture more abstract product design models, such as functionality or interface requirements. The captured models range from detailed function and behavior information to loose hierarchical artifact relationship views. This section begins by reviewing repository and knowledge-based systems and schemas developed to capture specific elements of design models. One such repository schema, developed by the National Institute of Standards and Technology (NIST), is reviewed and serves as a basis for design knowledge captured in the UMR repository system.

2.1.1 Knowledge-Based and Repository Systems. Researchers have built a variety of knowledge-based design information systems and have used different product models with varying degrees of abstraction. Although not all of the knowledge bases built were designed for the collection of function-based designed knowledge, information can still be extracted from these systems and their representations.

Summers [2] reports on a knowledge system designed for CAD-based feature elements with the intent of supporting designer activities. Although the modeling approach between the described CAD-based feature system and a functional-based representation system are different, both capture relevant design information. The CAD-based knowledge system from Summers exemplifies that all design knowledge is relevant dependent upon the domain, representation, and level of abstraction. Dixon [3] makes the point that feature descriptions and representations must be valid within their respective domain of use.

Architects, like designers, also have the need to store valuable design knowledge. Cherneff [4] describes an architectural knowledge-based system; even though this system may seem very distant from function-based knowledge systems, the underpinning philosophy in both is to store design information for reuse and collaboration. Cherneff explains that design abstractions all require context and that objects only make sense in the object schema in which they are defined. Again, he is echoing the need to develop design representations that are relevant to specific domains.

Functional representations have been used to represent design knowledge in early repository-like systems. One of these early systems, described by Sturges [5], used a block diagram approach based on “function logic” and was powered by Hypercard stacks to navigate function diagrams. The representation schema used by Sturges 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 on how a product operates functionally.

In hopes of providing guidance to product representation research, NIST has developed a set of information models to be used for modeling product knowledge at varying levels of detail. There are several data entities that 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 [6]. Along with these classes there is a set of specific information needed with each item and a specified type of value that can be entered.

All of the above NIST-identified design knowledge models exceed the representational capabilities of current commercial function-based computational design tools. A more flexible and vendor-neutral data structure is needed to represent the heterogeneous knowledge that designers use. The NIST Design Repository Project was initiated to meet this need [7–12]. 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 data format. Within the NIST XML code there are five different sections: Artifacts, Functions, Forms, Behaviors, and Flows. These sections contain information relative to their denoted naming system.

2.1.2 Product Data Management Systems. In recent years 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 explore a PDM system and demonstrate many uses of such a system [13]. The PDM system demonstrated collects requirements, functions, concepts, and part structures as well as property models. Additionally the PDM system stores the entire product structure, variants, revisions, documentation, and CAD models. Although function structures and property models can be stored within a PDM system, they are not readily capable of storing the detailed function-based information that is desired within the scope of this research. Gearing a PDM system to capture detailed functionality and using such a system to generate useful design tools would require heavy modification. A PDM system is geared toward use in the manufacturing side of product development and is fundamentally different from a design repository system to support conceptual design activities. Within the repository, similar pieces of design knowledge like CAD models and artifact hierarchies are stored and developed; however, the main focus is the mapping between functions and components and the compatibility of components to connect together as a system.

2.1.3 Commercially Available Systems. There is currently no product on the market that is truly a design repository; however, there are several packages that contain elements of a design repository. Such computerized design packages can be grouped into three basic categories: (1) mechanical computer-aided design (MCAD) packages that augment traditional CAD models with more abstract design knowledge; (2) systems engineering toolsets which contain higher level design information that may be used to generate CAD models; and (3) systems modeling and simulation packages that have little or no interaction with traditional CAD packages. For the MCAD packages, the typical approach is to add layers of abstract design knowledge to the existing CAD model. For all categories, no standard language has evolved, though there is widespread use of the process of functional decomposition.

Unigraphics—UG/WAVE (www.ug.eds.com/ug/). UG/WAVE is a MCAD package that adds abstract product design knowledge capability to the core Unigraphics CAD (or solid modeling) package. Product architecture information in a parametric product layout is captured in a “control structure.” This appears to be similar to a functional modeling approach to product design, but is more form oriented. The module allows “what-if” evaluation of simplified design alternatives, making the necessary modifications to the rest of the design as necessary. The package also stores subsystem design knowledge such that it can be reused in future products.

3SL—CRADLE (www.threesl.com/). CRADLE is a British systems engineering toolset composed of six modules that can be used together or separately. The systems modeling component offers robust support of several modeling notations, including functional block diagrams, behavior diagrams, and object-oriented support.

Table 1 Functional basis function terms

Primary	Branch	Channel	Connect	Control Magnitude	Convert	Provision	Signal	Support
Secondary	Separate	Import	Couple	Actuate	Convert	Store	Sense	Stabilize
	Distribute	Export	Mix	Regulate		Supply	Indicate	Secure
		Transfer		Change			Process	Position
		Guide		Stop				

Design knowledge is stored in a single repository structure accessible by all modules. CRADLE also allows high levels of reuse among subassemblies stored in its repository. While it is not a CAD package per se, it can export information to a variety of CAD formats.

Nu Thena Systems—FORESIGHT (www.nuthena.com). FORESIGHT is strictly a systems modeling and simulation tool. It takes a hierarchical approach to functional and architectural modeling, allowing as many levels of abstraction as desired. In addition to the functional and architectural model, a mapping between the two is stored as design knowledge. This provides a strong link between function and form design. The package does not interact with other CAD systems, and the functional language used favors electronic systems.

The concept of a design repository is extremely useful in the context of automated design storage and retrieval packages. Although design repository-based systems do not exist in commercial form today, the review of current commercial offerings indicates that elements of the design repository concept are being adapted by mainstream commercial product development systems, and that industry is moving towards the vision of design repositories. No clear direction exists for its development, though. A standard repository structure supported by fundamental functional and architecture modeling research is needed to guide work in this area.

2.2 Function Classifications. The ability to classify function is a key element in design models for the conceptual design phase [1,14]. Identifying product functionality from a defined taxonomy or vocabulary allows artifact information to be computed and parsed. Chiang conducts a thorough review of function definitions and the role of functionality in design support tools [15]. Reviewing early function-based engineering design research such as Miles [16], Rodenacker [17], and Pahl and Beitz [14], the first instances of using verb-noun pairs and input-output flow transformations for function representation are summarized. Miles developed functional representation on the premise that any product is useful because of the product’s inherent functionality. Rodenacker used descriptions of material, energy, or information along with an input-output transformation to represent functionality. From these early representations of functions, many researchers have continued work to accurately describe product functionality. One such continuation of functional representation is the functional basis.

2.2.1 The Functional Basis. Addressing the need for a clear vocabulary to describe product function, the functional basis has emerged as a standardized design language [18]. It was formulated in concert with NIST to unify two similar, independent research efforts aimed at defining a standard vocabulary from pre-

vious worldwide research efforts [12,19]. The functional basis consists of two sets of terminology: one containing action verbs to describe function and a second containing nouns to describe flow. The functional basis spans all engineering domains while retaining independence of terms. The function set of the basis is broken down into eight categories termed the primary classes. These classes have further divisions called the secondary and tertiary levels that offer increasing degrees of specialization. The primary class represents the broadest definition of distinct function, while the tertiary class provides a very specific description of function. The secondary level of the function set, containing twenty-one action verbs, is the most often used class of the basis. The primary class and secondary function terms are shown in Table 1.

The flow set of the functional basis allows for the associated function’s input and output flows also to be described. Similar to the function set, there are three distinct classes within the flow set of the functional language. Within the primary class of the flow set, there are three main categories used to describe flow: material, signal, and energy—as popularized by Pahl and Beitz [14]. Each of these categories has the capability to represent the input or output of a function. The secondary class of this set has 20 nouns that are used to describe the type of flow. It is the secondary class of this basis that is primarily used when describing a product. The primary class and secondary flow terms are shown in Table 2. The tertiary level is omitted from Tables 1 and 2 for reasons of brevity, and can be found in Ref. [18].

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 [20].

2.3 Design Models and Tools. Within the field of function-based engineering design, many methods that produce design tools have emerged. Below, we summarize a select few such design tools that support conceptual design activities.

Bill of materials (BOM): A bill of materials is a detailed description of all of the artifacts within a given product. This provides an easy way for designers to see a simple breakdown of parts contained within a given product. Although artifact function descriptions are not traditionally used as part of BOM representations, in the context of research activities at UMR, such descriptions—represented as function and flow pairs—are associated with each artifact. For example, the artifact motor has the functional description of “convert electrical energy to mechanical energy.” Additional information about an artifact’s mass, dimensions, manufacturing processes, or material composition is also recorded. The BOM is usually represented in a tabular format with

Table 2 Functional basis flow terms

Primary	Material	Signal	Energy		
Secondary	Human	Status	Human	Electrical	Mechanical
	Gas	Control	Acoustic	Electromagnetic	Pneumatic
	Liquid		Biological	Hydraulic	Radioactive
	Solid		Chemical	Magnetic	Thermal
	Plasma				
	Mixture				

Table 3 BOM for an electric coffee grinder

Part #		Qty.	Part Description	Part Color	Function(Sub-fct. Description)	Description	Physical Parameters	Mfg. Process
A1	1	2	Screws (2)		Secure weight	Mild steel screws	Dia=0.25 "	Rolled
A2	1	1	Bottom plastic disc	Black	Distribute M.E.	Black plastic piece	Dia=3.75 ", height=1 "	Injection Molding
	2	1	Plug and cord	Black	Import electricity, transfer electricity	-	Input 110V, 1-12 A	
A3	1	1	Electric motor	Various	Convert E.E. to M.E.	AC motor	110 Volt	OEM
A4	1	1	Metallic cup	Gray	Import solid, Guide solid, store solid	Drawn metallic cup	Dia=3 ", height=1.25 "	Drawing
	2	1	Grinding blade	Gray	Change solid, transfer M.E.	twin blade, steel	Length=2.5 " total	Casting
	3	1	Shaft cap (plastic)	White	Secure M.E.	Plastic, conical in shape	Height=0.25 "	Injection Molding
A5	1	1	Plastic-Lever operated Switch mechanism	White	Actuate electricity	White plastic piece	Height=4.8 ", width=0.25 "	Injection Molding
	2	1	External case/body	Black	Import M.E., Distribute ME, stabilize M.E.	Black cylindrical plastic piece	Dia=3.3 ", height=5.4 "	Injection Molding
A6	1	1	Top transparent Lid	Transparent	Import solid, export solid, export signal	Transparent plastic with knob to activate switch mechanism	Dia=3.2 ", height=2.5 "	Injection Molding

M.E. - Mechanical Energy E.E. - Electrical Energy

the artifacts or part number listed in the left-most column, where each row represents a different artifact. A partial BOM for an electric coffee grinder is shown in Table 3.

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 [19]. 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. An example functional model and picture of a Braun electric coffee grinder are shown in Figs. 1 and 2, respectively.

In addition to the subfunctions describing the conceptual functions of a product, supporting functions, formally defined as functions that describe manufacturing, assembly, and support features present in the embodied form of a product [21], are needed to completely describe an existing product's function. Supporting functions are used within the repository representation to augment design knowledge representation. When included in the design knowledge schema, supporting functions allow for accurate generation of design structure matrices by fully defining embodied

artifact interactions.

Function component matrix: A function-component matrix records the component(s) that solve each function. Within the matrix, columns designate product components and rows designate the subfunctions of the product. For a single component, the matrix is binary, with a "1" showing that the component solves the corresponding function and a "0" indicating no relationship. When multiple product function-component matrices are aggregated together (known as a chi-matrix) the function component can be used to generate concepts [22]. The function-component matrix also serves as a roadmap linking the functional model and bill of materials. An example function-component matrix is shown in Table 4.

Design structure matrix: The design structure matrix (DSM) is a matrix in which rows and columns represent the interaction of artifacts within a product [11]. When two artifacts within a product interact with one another in some way, the cell where a row and column corresponding to those two artifacts meet is marked with a "1" (or alternatively an "X"). Cells corresponding to pairs of artifacts that do not interact are marked with a "0" (or alternatively left blank). In the simplest DSM representation, the matrix is symmetric matrix because the interaction between artifacts A

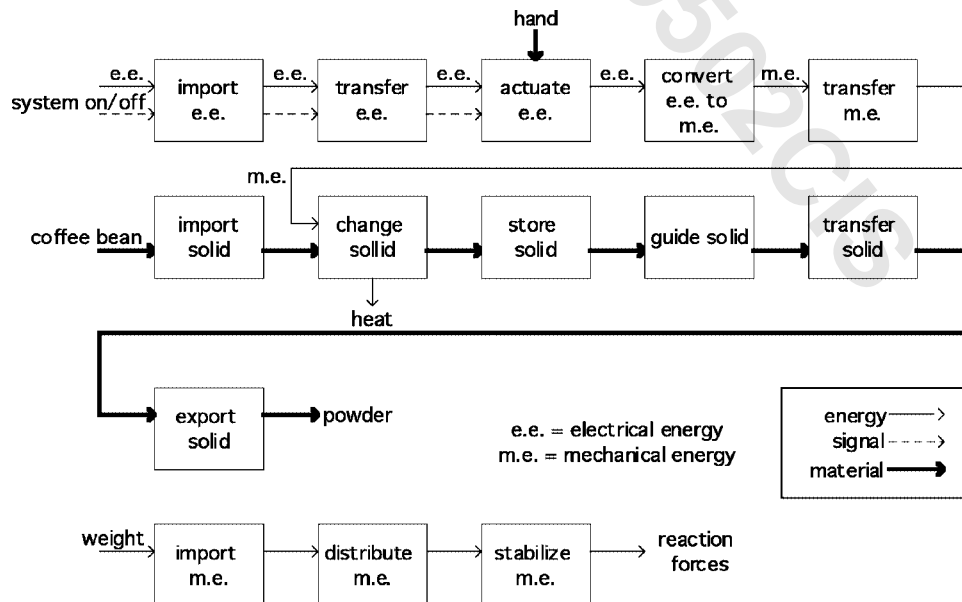


Fig. 1 Functional model for an electric coffee grinder



Fig. 2 Braun electric coffee grinder

and B will show up at the intersection of row A and column B, as well as at the intersection of row B and column A. This is useful in the design process to see how artifacts within a product relate to each other physically. Table 5 shows a DSM of an electric coffee grinder.

3 Research Problem

The design repository effort at UMR has been in existence since the summer of 2000 and had over 11 different researchers contributing product data. Initially recorded product data took the form of customer needs lists, bills of materials, functional models, module-based functional models, function-component matrices, design structure matrices, product vectors, artifact photos, and as-

Table 4 Function component matrix of an electric coffee grinder

Function-Component Matrix for Braun Coffee Grinder	bottom plastic disc	electric motor	external case	grinding blade	metallic cup	plastic lever operated switch	plug and cord	screws	shaft cap	top transparent lid
distribute mechanical energy	1	0	0	0	0	0	0	0	0	0
guide mechanical energy	1	0	0	0	0	0	0	0	0	0
secure mechanical energy	1	0	1	0	0	0	0	0	1	0
change electrical energy	0	0	0	0	0	0	0	0	0	0
export solid material	0	0	0	0	0	0	0	0	0	1
import solid material	0	0	0	0	0	0	0	0	0	0
convert electrical energy	0	1	0	0	0	0	0	0	0	0
guide electrical energy	0	1	0	0	0	0	0	0	0	0
secure electrical energy	0	1	0	0	0	0	0	0	0	0
import mechanical energy	0	0	1	0	0	0	0	0	0	0
stabilize mechanical energy	0	0	1	0	0	0	0	0	0	0
change mechanical energy	0	0	0	1	0	0	0	0	0	0
guide solid material	0	0	0	1	1	0	0	0	0	0
transfer solid material	0	0	0	1	1	0	0	0	0	0
store solid material	0	0	0	0	1	0	0	0	0	0
actuate control signal	0	0	0	0	0	1	0	0	0	0
guide control signal	0	0	0	0	0	1	0	0	0	0
import electrical energy	0	0	0	0	0	0	1	0	0	0
transfer electrical energy	0	0	0	0	0	0	1	0	0	0
couple mechanical energy	0	0	0	0	0	0	0	1	0	0
couple solid material	0	0	0	0	0	0	0	1	0	0
secure solid material	0	0	0	0	0	0	0	1	0	0
import status signal	0	0	0	0	0	0	0	0	0	1

Table 5 Design structure matrix of an electric coffee grinder

Braun Coffee Grinder Hand Generated DSM	bottom plastic disc	electric motor	external case	grinding blade	metallic cup	plastic lever switch	plug and cord	screws	shaft cap	transparent lid
bottom plastic disc	1	1	1				1	1		
electric motor	1	1		1	1	1	1	1	1	
external case	1		1			1				1
grinding blade		1		1	1				1	
metallic cup		1		1	1					
plastic lever switch		1	1			1				1
plug and cord	1	1					1			
screws	1	1						1		
shaft cap		1		1					1	
transparent lid			1			1				1

sembly instructions. Most of these elements were created inside of individual spreadsheets and drawing applications. To view the product data a user would navigate a folder structure to an individual product (see http://function.basiceng.umn.edu/Repository_Data). Within a single product folder, additional folders were presented allowing the user to further navigate to a specific design tool. Often there would be multiple versions of a single design representation for a single product. Once inside the appropriate folder the user could then open and view the desired design information.

The most noticeable issues with the previously archived product data were the inconsistency of product representation language, format, and the difficulties with retrieving archived design knowledge. The inconsistency between artifact representations can be linked to the time span and number of researchers associated with the product data collection. Producing design tools from the file hierarchy-based repository proved to be a tedious process. For example, an aggregate function component matrix was populated manually by dragging and dropping individual product function component matrices into a combined function component matrix.

From working with this early design knowledge archive, the need, and thus the research problem, was to create a coherent system capable of accurately representing product design knowledge and effectively supporting design knowledge reuse. The envisioned repository system consisted of 3 distinct elements: (1) a database; (2) a data input application; and (3) a web portal to support knowledge reuse. Specifically, the steps to create such a system included (i) determining an appropriate product design knowledge representation schema; (ii) developing a prototype repository for product data archival; (iii) testing compatibility and integrity of the prototype repository; and (iv) implementing the revised repository system. The three elements of the research problem are treated separately in the following sections.

4 The Repository Database

4.1 Repository Knowledge Representation. As a first step towards building a new design repository, existing artifact information was reviewed to better understand what attributes should be included in the future to ensure accurate representations. Functional descriptions of products, a key component of the repository system, were modeled at various levels of detail (i.e., primary, secondary, or tertiary). According to Hirtz et al. [18] the secondary level of function is the preferred level for artifact representation. Another issue encountered was inconsistencies in the use of abbreviations for functions, such as “Ex” for export, “Im” for import, “Sep” for separate, or “Dist” for distribute. Recorded artifact physical parameters such as dimensions, material, and manufac-

Table 6 Repository data fields

Data Fields	Type	Relational	List Value
Artifact Name	Text		
Part Family	Text		
Part Number	Numerical		
Sub Artifact of	Text	X	
Quantity	Numerical		
Description	Text		
Artifact Color	Text	X	
Component Naming	Text	X	X
Assembly	Boolean		X
Supporting Function	Boolean		X
Input Artifact	Text	X	
Input Flow	Text	X	X
Subfunction	Text	X	X
Output Flow	Text	X	X
Output Artifact	Text	X	X
Physical Parameter Type	Text	X	X
Physical Parameter Value	Numerical		
Physical Parameter Metric	Text	X	X
Manufacturing Process	Text	X	X
Material	Text	X	X
Primary Identifier	Text	X	X
Failure Mode	Text	X	X
Risk Element	Text	X	X

turing processes varied greatly as well. Most artifact dimensions were represented by “L=6, W=2, H=3.” Noticeably the lack of associated units of measure results in an imprecisely specified product description. Multiple artifact dimensions were often keyed into a single cell rather than individual cells, with no consistent use of labeling. Similar issues were present with material and manufacturing process data.

Another hurdle to tackle was deciding how much textual product information would be necessary to automatically generate a functional model, a function component matrix, and other representation types described in Sec. 2.3. The functional model of a given product includes subfunctions as well as input and output flows. In order to represent this type of information digitally, both input and output flow must be recorded, along with the given artifact’s subfunction. These artifacts and associated subfunctions generally have input and output flows linking them to other artifacts and subfunctions, but can also interface with an environment outside of the functional model (import/export). In the repository’s initial state, this type of artifact linking had not been captured. For an initial test implementation it was decided that firmly establishing a link between a subfunction’s input and output flow would capture design knowledge necessary to support the generation of FCMs, DSMs, and BOMs.

Returning to important customer needs, the repository system needs to be robust enough to handle many product domains while maintaining product data in an interchangeable format. To achieve this goal, a robust data schema is defined for knowledge capture based on the original Design Repository Project at NIST [7] and defined vocabularies and procedures by which to enter data are selected. The different representations chosen for this work are based on product information flow schemes [10] and dissection processes [1,23] as well as information required in modern design methods and tools such as functional representation languages [18], component naming schemes [22,24,20]. A listing of the predominant types of design knowledge captured in the repository system and its data type is shown in Table 6 denoted by the data fields column.

Artifact name: a free-form text field where the user can define the name of an artifact. Part family: a free-form text field that can be used to catalog similar artifacts as a type or family. Part number: numbered artifact field that sequentially numbers artifacts as they are populated. Subartifact of: a text field that creates an artifact hierarchy by establishing a parent-child relationship. Quantity: a user input numerical value to denote the quantity of a particular artifact. Description: a free-form text box that allows the user to further describe an artifact. Artifact color: a free-form and relational field that allows the color of an artifact to be stored. Component naming: a text field that pulls from a list populated

with standard component naming terms [25]. Assembly: a Boolean value used to denote whether an artifact is singular or a combination of more than one artifact. Supporting function: a Boolean value that denotes whether the artifact subfunction is supporting or conceptual [21]. Input artifact & output artifact: textual fields used to trace flow from the current artifact to the corresponding input and output artifacts. Input flow & output flow: a list value (from the functional basis) that traces the input and output flow of an artifact. Subfunction: a list value (from the functional basis) that defines the actual function of an artifact. Physical parameter type, value, & metric: used to define a rough geometry of an artifact. Manufacturing process: a list value used to denote the type(s) of manufacturing process(es) used to produce the artifact. Material: a list value used to denote the predominant material of an artifact.

4.2 Software Options. For the data storage portion of the suite, four technologies were reviewed; MYSQL [26], POSTGRES [27], ENTERPRISE JAVA BEANS [28] and ORACLE [29]. Initially five choices for web display were identified; ASP (Active Server Pages) [30], PHP (Hypertext Preprocessor) [26], JSP (Java Server Pages) [31], COLDFUSION [32], and STRUTS [33]. Four technologies were also reviewed for the standalone entry application; JAVA [34], C/C++ [35], Microsoft’s NET [36], and REALBASIC [37]. The pros and cons of each software technology are summarized in the form of a modified morphological matrix (Table 7).

The pros and cons of available software components were evaluated against a set of initial customer needs. The repository team set out to develop software that would operate independent of the type of operating system. Because of this high-level need, any software technologies listed in Table 7 that are platform dependent or proprietary were removed from consideration. The repository team also wanted to minimize the time for development; thus, software technologies that would require longer development times or have a steep learning curve were removed to further reduce the number of options. Software technologies that would require a significant monetary investment were also eliminated from the list of possibilities. Another desire was the ability to reuse code objects between the web display and entry application—this pointed to JSP and JAVA, respectively. Finally, the choice was made to use POSTGRES for data storage, JSP for web display, and JAVA for the entry application.

4.3 Implemented Database. POSTGRES, a variant of MySQL, is a relational database package that provides a good level of scalability and a number of development tools and packages. POSTGRES was chosen over MYSQL because of scalability and for supporting user-defined data types. Like most database packages, POSTGRES uses tables to represent data and relationships. Applying the determined repository fields to the POSTGRES database created an intricate table structure. The main repository table within the POSTGRES database is the Artifact table. Nearly all of the remaining pieces of product design information are connected to the Artifact table in some way. Figure 3 shows the Artifact table along with the Subfunction_Type, Subfunction, Artifact_Flow, and Flow tables.

Each row in the database table denotes a field within the database. The first column in the database table is the name assigned to that data field. The second column in the database table denotes the particular data type for the corresponding field. Data types can take the form of serial, integer, Boolean, or characters (varchar). Within the data table, the third column can be designated by pk, fk, or alternatively be left blank. Using pk denotes that the field is a primary key in the table and fk denotes the field has a foreign key that references another table. A primary key unambiguously specifies a row of a table because primary keys must be unique across all tables within the database. A foreign key (fk), creates a relationship between a row in one table to a row within another table. Usually a foreign key is used as a pointer to a primary key field of another table within the database. Foreign keys are the

Table 7 Software options summary

	Technology	Pros	Cons
Data Storage	MySQL	fast (read)	no stored procedures
		well supported	no views/rollbacks
		easy	no fk constraints
		JDBC/ODBC Support	
	PostgreSQL	scalable	slower than MySQL
		object-related inheritance	difficult administration
		JDBC/ODBC Support	xml native support
		have viewers and cursors	
	Enterprise Java Beans	user defined data types	
		robust	harder to develop
		scalable	requires an application server
		supports transactions	learning curve
Oracle	supports searching		
	easy to create relationships		
	fast		
	incredibly scalable	licensing	
Web Display	ASP	fast	steep learning curve
		full transaction support	harder to administer
		product support	
		easy	platform dependent
	PHP	well supported	
		easy to administer	
		fast on IIS	
		easy	maintainability
	JSP	platform independent	
		object oriented	
		can use parts of code for standalone	
		custom tags	
ColdFusion	good xml/xslt support		
	easy to learn	proprietary	
	easy to use	expensive	
	engine is platform independent	difficult to maintain	
Struts	can extend functionality with C++		
	declarative coding	learning curve	
	based on well known design pattern		
	scalability		
Entry Application	Java	platform independent	GUI building
		can reuse objects from JSP	slower
		support	
		can use Java RMI/EJB	
	C/C++	easy data retrieval	
		easy storage and searches	
		very fast	not truly platform independent
		well supported	possibly no GUI (depending on platform)
	.net	standardized	
		good XML parsing support	
		easy to learn	platform dependent
		support	slow
RealBasic	easy to interface with asp	bloated	
	easy to build interfaces	requires .net to be installed on client machines	
	good xml parsing support		
	easy to learn	no good xml support	
		good user interface support	
		somewhat platform independent	

main elements of any relational database because they are the operators that establish relationships between sets of data. The arrows within a database are used to denote table relationships. For example, the Artifact_Flow table has three arrows emerging from it that terminate at the Artifact table (Fig. 4). These arrows describe relationships between foreign keys within the Artifact_Flow table to a primary key within the Artifact table.

5 Product Knowledge Data Entry

To populate the database, a single application to enter all of the identified product models was created. A commercial database application, FILEMAKER PRO, was selected for the first generation entry application because of its cross-platform capability and relative simplicity of operation compared to other heavy-duty commercial database offerings. The entry application, called the enhanced bill of materials (EBOM), handles entry, management, and

export of repository knowledge to the POSTGRESQL database. The entry method parallels the original repository editor of the NIST Design Repository Project [12] in that design knowledge is entered on an artifact by artifact basis. As with the NIST project, each artifact may itself be composed of additional subartifacts. Thus, to accurately represent product design knowledge, an extensive representation for each artifact must be created. The layout includes the standard elements of a BOM, along with additional information regarding the flow paths so that an accurate digital representation of the product can be achieved. A screenshot of the current EBOM application is shown in Fig. 5.

To illustrate product design information entry, consider the functional model snippet shown in Fig. 4 and taken from the electric coffee grinder functional model (shown previously as Fig. 1). The ellipses represent artifacts as recorded by the EBOM application. In this example, the *plug and cord* artifact is the artifact

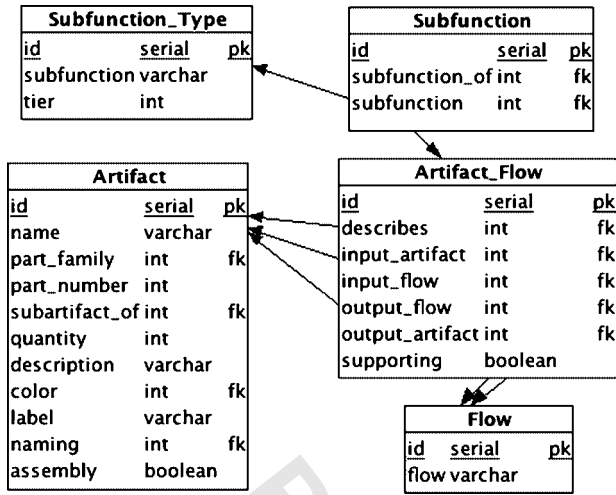


Fig. 3 Main database tables

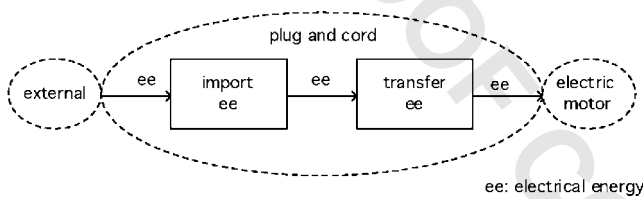


Fig. 4 Functional model snippet of electric coffee grinder

being represented. Directly, the *plug and cord* has the subfunctions of importing and transferring electrical energy. The source of the electrical energy is considered outside of the bounds of the electric coffee grinder and is denoted as an *external* artifact. The *electric motor* is the destination of electrical energy leaving the *plug and cord*.

Table 8 shows the functional model of the *plug and cord* from Fig. 4 translated to the EBOM subfunction representation schema. The columns in the table of input artifact, input flow, subfunction, output flow, and output artifact are in the same layout as the EBOM application. Following the flow of electrical energy from left to right, starting with the first row, electrical energy is imported from *external* and then output to *internal*. The artifact *internal* is used to chain subfunctions together that exist within the same artifact. Cases where *internal* is used as both the input and output artifact are often found in artifacts where the material, signal, or energy flow are changed several times within the artifact. *Internal* is then used as the input artifact in the second row of the table, meaning that the same electrical energy that has been imported by the first subfunction is now linked to the second subfunction of transfer. Piecing the rows together, electrical energy is imported from an *external* artifact and then transferred to the *electric motor*. Figure 5 shows the corresponding EBOM entry screenshot of the *plug and cord* artifact.

A test bed of ten consumer household products was chosen to test the initial implementation of the EBOM. These ten products were chosen because they contained artifact representations spanning nearly all classes of flows and subfunctions and were generally electromechanical consumer products. Having a test dataset that maps into each of the function and flow types helped ensure that the most possibilities and combinations of artifact setup were taken into consideration while developing a new repository entry method.

Fig. 5 Repository artifact input screen

Table 8 EBOM subfunction mapping for plug and cord artifact

Input Artifact	Input Flow	Subfunction	Output Flow	Output Artifact
external	electrical energy	import	electrical energy	internal
internal	electrical energy	transfer	electrical energy	electric motor

To eliminate discrepancies in language representation, abbreviations, and formatting, defined lists of commonly used BOM elements along with a structured set of the functional language terms must be implemented. This is achieved through the use of master lists within a relational database. This ensures that all products and artifacts represented across the entire span of repository data are consistent, not only in language but also in format. For example, the material and manufacturing process fields are limited to a predefined list of commonly used materials and manufacturing techniques. The initial material master list, shown in Table 9, was created by examining the most commonly used materials that were already represented within the repository. The manufacturing process list shown in Table 10 was adapted from the Dixon and Poli [38] Taxonomy of Manufacturing, and was combined with commonly seen processes from previous product dissection. Master lists were also created for flow and function secondary classes

Table 9 Material type master list

Material
Plastic
ABS
Nylon
Aluminum
Steel
Rubber
Wood
Concrete
Composite
Metal
Foam
Glass
Iron

Table 10 Manufacturing processes master list

Manufacturing Process
Injection_molding
Stamping
Extrusion
Rolling
Casting
Forging
Machining
Forming
OEM

using the Hirtz et al. [18] functional basis. Note that a user can change the master lists within the EBOM application but only a system administrator can update a master list within the POSTGRESQL database. This means that only data in the correct specified form can be imported from the EBOM application to the database.

The EBOM application data types and fields match those of the POSTGRESQL database and are used for repository data input. Design knowledge populated in the EBOM application is extracted from the FILEMAKER PRO database through a droplet application. The droplet application parses through a particular product file, connects to the POSTGRESQL database, and finally writes the new data. The EBOM application uses the same relational lists and constraints as described in Secs. 4.1 and 4.3. EBOM templates are available for those wishing to contribute design knowledge to the repository.

6 Design Knowledge Reuse Tools

Design tool generation and data processing for web display is handled by the server and database side of the repository. When generating design tools, the database runs a fairly simple script to produce the desired tool's result. For example, the Function Component Matrix is generated by only a few lines of code executed by the database. Shown below in pseudocode are the basic steps necessary to create a FCM within the repository.

The first task is to find all of the artifacts in the database for the given system.

```
artifacts=findArtifacts( )
```

Next, all of the functions that a given artifact performs must be found.

```
For each artifact found
```

```
Artifact a=currentArtifact
```

```
For each subfunction
```

```
For each inputflow
```

```
String fn=artifact.subfunction+artifact.inputflow
```

```
if(functions!=contain(fn))
```

```
functions.add(fn)
```

The matrix must then be created.

```
int rowSize=functions.size( )
```

```
int colSize=artifacts.size( )
```

```
matrix=new int[rowSize][colSize]
```

```
For each row
```

```
String function=currentArtifact
```

```
For each column
```

```
Artifact artifact=currentArtifact
```

```
if(artifact.performsFunction(function))
```

```
matrix[row][col]=1;
```

```
else
```

```
matrix[row][col]=0;
```

The same basic process for FCM generation is also used to generate Design Structure Matrices. For DSM generation the routine operates on input and output artifacts as opposed to an artifact subfunction.

The repository web front end was constructed using JSP. JAVA Server Pages is a web technology used to create dynamic web applications that separate content from the overall site design layout. The separation of content from design and component-based foundation allows for faster web application development and easier maintainability. Component-based design is inherent in JSP since it is an extension of JAVA servlet technology, which reduces overall programming complexity by separating key components of the repository system into objects. Objects are essentially a

Design Engineering Lab REPOSITORY Artifact Browse

System: braun coffee grinder

Artifact Name: electric motor
Part Family: Not Specified
Part Number: 4
Sub Artifact Of: external case
Quantity: 1
Description: provides mechanical motion
Artifact Color: various
Component Naming: Not Specified

Artifact Photo:

Input Artifact	Input Flow	Subfunction	Output Flow	Output Artifact
plastic lever operated switch	electrical energy	convert	mechanical energy	grinding blade

Supporting Functions

shaft cap	solid material	guide	solid material	shaft cap
metallic cup	solid material	secure	solid material	metallic cup

Physical Parameters: No Parameters Specified
Manufacturing Process: Material: metal; No Process Specified
Primary Identifier:
Secondary Identifier:
Failure Mode:

Component Relation Equation: Not Specified
Equation Metric:

Variable	Variable Name	Variable Value	Variable Metric
No Variables Specified			

Home | Browse Systems | Search | Design Tools | Log Out

Fig. 6 Screenshot of browse feature

collection of related attributes and methods that can be accessed or manipulated from outside the object. Objects are a fundamental concept in JAVA, an object-oriented programming language.

JAVA Server Pages was chosen above other languages for the web interface because of its object-oriented design, and because it has the ability to interact with JAVA objects. The objects used for the web interface have been developed for a variety of tasks that include ways to manage connections to the database and artifact information retrieval. Since these objects are written in JAVA and are neither platform dependent nor require server specific modifications, these objects can be reused in other parts of the repository system, namely the client application. The reuse of interoperable code greatly reduces development time for both the web interface and the stand-alone client application.

When an artifact is viewed with the web interface, the JSP creates a JAVA object based on a unique identifier for each artifact that corresponds with a unique key in the database. The object automatically calls other objects for connecting to the database, executing database queries, and retrieving other pertinent information. Such information can include the artifact's name, description, quantity, and other attributes. Once an object for an artifact is created, the web interface then obtains the artifact's information by using function calls to retrieve the applicable information. These data are then output to the webpage with the artifact's information. The same process holds true for the other artifact information such as flow, parameters, and the image associated with the artifact.

The repository web interface, which offers guest and registered user access is located at <http://function.basiceng.umn.edu/>

repository/. The top-level options within the web repository are: browse, search, design tools, design methodology dictionary and account information. With the web-based repository a user can browse and search artifacts, generate design tools, and view a dictionary of function and flow terms.

The browse feature allows users to navigate through the repository. When browse is initially selected, all of the high-level systems within the repository are shown at the left of the screen. The systems can be expanded such that artifacts within the system are exposed. A hierarchical menu system allows for systems to be expanded through subassemblies down to singular artifacts. The menu system draws information from the Subartifact_Of field of the database to establish artifact hierarchy. Finally, when an artifact or assembly is selected, a repository listing of the artifact is shown on the right portion of the screen. A screenshot of the browse feature is shown in Fig. 6.

The repository search feature allows for users to search on a string of input flow, subfunction, and output flow. Once the search criteria are selected, the database is then queried. Figure 7 shows how a search string is input and how search results are returned. The returned search results are hyperlinked to the browse page for that particular artifact.

When a repository user selects the design tool option, they are presented with a listing of the high-level systems contained in the repository and selection boxes to denote the type of desired design tool output. A user can select single or multiple systems for design tool generation. Once the systems are selected, a summary of the selected systems is presented, notifying the user of the number of artifacts within the systems and system descriptions. The reposi-

Design Engineering Lab REPOSITORY Artifact Search

Search for Artifacts that meet the following criteria:

electrical energy convert mechanical energy Search

Design Engineering Lab REPOSITORY Artifact Search

Search Results:

- electric motor
- electric motor
- electric motor
- motor
- electromagnet
- motor
- condensor motor
- rotor
- stator

Search took: **0.042** seconds.

Fig. 7 Search criteria selection and search results

Design Engineering Lab REPOSITORY Design Tools

- b and d electric knife
- b and d palm sander
- bissel handvac
- brake system
- braun coffee grinder
- digital scale
- dirt devil vacuum
- dremel engraver
- electric stapler
- ge fridge
- jigsaw
- rice cooker
- robotics
- salton electric wok
- star scanner

Artifact Summary

Name: braun coffee grinder

Description: Description not specified

Number of Artifacts: 11 artifacts

Design Tool Options

- FCM (functional component matrix)
- DSM (design structure matrix)
- BOM (bill of materials)
- Functional Model (Feature currently not implemented)
- Transfer Function Calculation (Feature currently not implemented)

Generate Tools

Multiple selections are allowed (cmd+click or ctrl+click)

Fig. 8 Design tool options

Table 11 Combined function component matrix

Combined Function-Component Matrix (Braun Coffee Grinder and Salton Electric Wok)	back cover	back cover screw	bottom plastic disc	electric motor	external case	front cover	grinding blade	handles	lid	lid handle	lid screw	metal cover	metal cover screws	metal rod	metal supporting stand	metallic cup	pan	plastic lever operated switch	plug and cord	potentiometer mechanism	screws	shaft cap	snaps	supporting stand screws	temperature knob	top transparent lid	wire snaps
secure mechanical energy	1	1	2	0	1	1	0	0	0	0	1	1	1	0	0	0	0	0	0	0	2	1	0	1	0	0	0
distribute mechanical energy	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
guide mechanical energy	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
change electrical energy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
export solid material	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
import solid material	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
convert electrical energy	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
guide electrical energy	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
secure electrical energy	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
import mechanical energy	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
stabilize mechanical energy	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
change mechanical energy	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
guide solid material	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0
transfer solid material	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
stop thermal energy	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
sense thermal energy	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
store solid material	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
distribute solid material	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
transfer thermal energy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
guide control signal	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0
actuate control signal	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
import electrical energy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0
transfer electrical energy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	1	0	0	0	1
regulate electrical energy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0
couple mechanical energy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
import status signal	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

tory can currently output function component and design structure matrices as well as bills of materials. The user can select to output any combination of the specific design tools. With the desired design tool output selected, clicking the “generate tools” button will display an additional page with links to each of the selected components. When the individual links are selected the server is queried and that particular design tool is generated. Because the design tools are not stored but rather pull directly from the repository database, the user will always be presented with the most up-to-date design tool. A screenshot of the design tools main page is shown in Fig. 8.

The outputs for the query shown in Fig. 8 are a function component matrix (Table 4) and a design structure matrix (Table 5). Multiple systems can be selected for design tool generation. Allowing multiple system selections enables for generation of an aggregated bill of materials or a chi matrix [22] for use in concept generation. An aggregated function component matrix (chi matrix) of the coffee grinder and an electric wok is shown in Table 11.

7 Conclusions

The repository project at UMR offers a significant impact 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 and offers extended capability to current product data management applications. The repository system supports automatic design tool generation and knowledge search and navigation. The prototype repository system, available on the web, reaches designers and researchers worldwide. Known users include GM, NASA Ames

Research Center, the Engineering Design and Optimization Group at Pennsylvania State University, Bucknell University, Virginia Tech, UT-Austin, and the University of Maryland-BC. All of the listed institutions use different aspects of the repository in accordance with their particular line of research.

The current repository system establishes a foundation for corporate technical memory storage. Knowledge from experts can be catalogued within product design knowledge by the repository. Through the EBOM entry application, product design knowledge is represented consistently and contained within a single database. In addition to consistent data representation, the EBOM is the only application needed for product design knowledge entry.

Automatic design tool generation is another benefit of this repository implementation. By allowing design tools to be created dynamically and built for any product or set of products, designers are no longer presented with the tedious task of generating these tools by hand.

The implementation of the prototype 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. With each additional design knowledge representation added, the repository emerges a smarter system. Over time, a mature repository could likely be used to support artificial intelligence systems.

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 corporate knowledge of human experts in a form that is parsable and computable. One such aspect involves identifying pertinent design knowledge attributes and separating those attributes from variables that can be considered noise. Expert knowledge can then be stored and used to support automated reasoning in fields such as artificial intelligence.

8 Future Work

Future work includes finalizing the stand-alone JAVA-based repository input application, increasing the number of design tools that can be directly exported from the repository, incorporating more knowledge representations, and further database and computational optimization. In particular, a desired design tool output is the ability to automatically export functional models containing not only conceptual level product information but also supporting level design knowledge. Scalable Vector Graphics (SVG) and XSLT (eXtensible Stylesheet Language Transformations) are being investigated to generate functional models directly from the design knowledge database. Another ultimate goal of the repository system is to build capabilities to support concept generation. With each additional representation or feature added to the repository, ways of optimizing system speed and efficiency must be explored in order to keep the repository as an effective design tool.

The repository team is working immediately to update previously extracted product design knowledge to include supporting function, component naming, and assembly representations. By augmenting all of the existing products with this design knowledge, the repository will provide more detailed design knowledge representation. Currently, the repository is populated with approximately 50 consumer products and it is desired to expand out of the consumer product realm to larger, more complex systems as well as single part representation. Long term, the intent is to populate the repository with thousands of products spanning multiple domains. With a foundation of numerous products, data mining can also be performed to further investigate relationships between design knowledge attributes. Through data mining, computational subroutines can further be refined to produce more accurate and solution viable results when performing search or concept generation operations.

Acknowledgments

This work is supported by the National Science Foundation under Grant DMI-9988817. 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] Otto, K., and Wood, K., 2001, *Product Design: Techniques in Reverse Engineering, Systematic Design, and New Product Development*, Prentice-Hall, New York.
- [2] Summers, J., Maxwell, D., Camp, C., and Butler, A., 2000, "Features as an Abstraction for Designer Convenience in the Design of Complex Products," DETC2000/CIE-14642, Proceedings of DETC-2000, Baltimore, MD.
- [3] Dixon, J., Libardi, E., Luby, S., Vaghul, M., and Simmons, M., 1987, "Expert Systems for Mechanical Design: Examples of Symbolic Representations of Design Geometries," *Eng. Comput.*, **2**(1), pp. 1–10.
- [4] Chermeff, J., 1991, "Communicating Design Representations: The Role of Interpretation," *Computer-Aided Engineering Journal*, **8**(4), pp. 153–159.
- [5] Sturges, R., O'Shaughnessy, K., and Kilani, M., 1996, "Computational Model for Conceptual Design based on Extended Function Logic," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **10**(4), pp. 255–274.
- [6] Fenves, S., 2001, "A Core Product Model for Representing Design Information," NISTIR 6736, National Institute of Standards and Technology, Gaithersburg, MD.
- [7] Szykman, S., Sriram, R., and Smith, S., 1996, "Proceedings of the NIST Design Repository Workshop," National Institute of Standards and Technology, Gaithersburg, MD.
- [8] Murdock, J., Szykman, S., and Sriram, R., 1997, "An Information Modeling Framework to Support Design Databases and Repositories," DETC97/DFM-4373, Proceedings of DETC'97, Sacramento, CA.
- [9] Szykman, S., Racz, J., and Sriram, R., 1999, "The Representation of Function in Computer-Based Design," DETC99/DTM-8742, Proceedings of DETC99, Las Vegas, NV.
- [10] Shooter, S., Keirouz, W., Szykman, S., and Fenves, S., 2000, "A Model For Information Flow in Design," DETC2000/DTM-14550, Proceedings of DETC2000, Baltimore, MD.
- [11] Szykman, S., Fenves, S., Shooter, S., and Keirouz, W., 2001, "A Foundation for Interoperability in the Next-Generation Product Development Systems," *Comput.-Aided Des.*, **33**(7), pp. 545–559.
- [12] Szykman, S., 2002, "Architecture and Implementation of a Design Repository System," DETC2002/CIE-34463, Proceedings of DETC2002, Montreal, Canada.
- [13] Svensson, D., and Malmqvist, J., 2001, "Integration of Requirement Management and Product Data Management Systems," DETC2001/CIE-21246, Proceedings of DETC2001, Pittsburgh, PA.
- [14] Pahl, G., and Beitz, W., 1996, *Engineering Design: A Systematic Approach*, Springer-Verlag, London, UK.
- [15] Chiang, W., Pennathur, A., and Mital, A., 2001, "Designing and Manufacturing Consumer Products for Functionality: A Literature Review of Current Function Definitions and Design Support Tools," *Integrated Manufacturing Systems*, **12**(6), pp. 430–448.
- [16] Miles, L., 1961, *Techniques of Value Analysis and Engineering*, McGraw-Hill, New York.
- [17] Rodenacker, W., 1971, *Methodisches Konstruieren*, Springer, Berlin.
- [18] Hirtz, J., Stone, R., McAdams, D., Szykman, S., and Wood, K., 2002, "A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts," *Res. Eng. Des.*, **13**(2), pp. 65–82.
- [19] Stone, R., and Wood, K., 2000, "Development of a Functional Basis for Design," *ASME J. Mech. Des.*, **122**(4), pp. 359–370.
- [20] McAdams, D., and Wood, K., 2000, "Quantitative Measures for Design By Analogy," DETC2000/DTM-14562, Proceedings of DETC2000, Baltimore, MD.
- [21] 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.
- [22] 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.
- [23] Sheppard, S., 1992, "Mechanical Dissection: An Experience in How Things Work," *Engineering Education Conference: Curriculum Innovation & Integration*, Santa Barbara, CA.
- [24] Pimpler, T., and Eppinger, S., 1994, "Integration Analysis of Product Decompositions," Proceedings of the ASME Design Theory and Methodology Conference, Minneapolis, MN., DE-Vol. 68.
- [25] 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.
- [26] Thomson, L., and Welling, L., 2001, *PHP and MySQL Development*, SAMS Publishing.
- [27] Douglas, K., and Douglas, S., 2003, *POSTGRES SQL*, SAMS Publishing.
- [28] Monson-Haefel, R., 2001, *ENTERPRISE JAVA BEANS*, O'Reilly & Associates, Sebastopol, CA.
- [29] Beaulieu, A., and Mishra, S., 2002, *Mastering ORACLE SQL*, O'Reilly & Associates, Sebastopol, CA.
- [30] Weissinger, A. K., 2000, *ASP in a Nutshell*, O'Reilly & Associates, Sebastopol, CA.
- [31] Murach, J., and Steelman, A., 2003, *Murach's JAVA Servlets and JSP*, Mike Murach & Associates, Fresno, CA.
- [32] Bainum, S., Camden, R., and Rish, G., 2002, *Mastering COLDFUSION MX*, Sybex.
- [33] Cavaness, C., 2003, *Programming JAKARTA STRUTS*, O'Reilly & Associates, Sebastopol, CA.
- [34] Deitel, H. M., and Deitel, P. J., 2003, *JAVA: How to Program*, Prentice-Hall, Englewood Cliffs, NJ.
- [35] Deitel, H. M., and Deitel, P. J., 2002, *C++ How to Program*, Prentice Hall, Englewood Cliffs, NJ.
- [36] Balena, F., 2002, *Programming Microsoft Visual Basic. NET*, Microsoft Press.
- [37] Neuburg, M., 2001, *REALBASIC: The Definitive Guide*, O'Reilly & Associates, Sebastopol, CA.
- [38] Dixon, J., and Poli, C., 1995, *Engineering Design and Design for Manufacturing: A Structured Approach*, Field Stone, Conway, MA.