The Pennsylvania State University

The Graduate School

College of Engineering

COMPUTATIONAL TOOLS TO IMPROVE INNOVATION AND CREATIVITY IN ENGINEERING DESIGN

A Thesis in

Mechanical Engineering

by

Vamsy Godthi

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

May 2010

The thesis of Vamsy Godthi was reviewed and approved* by the following:

Cari R. Bryant Arnold
Assistant Professor of Engineering Design & Mechanical Engineering
Thesis Advisor

Timothy W. Simpson
Professor of Mechanical & Industrial Engineering

Karen A. Thole
Professor of Mechanical Engineering
Head of the Department of Mechanical and Nuclear Engineering

*Signatures are on file in the Graduate School

ABSTRACT

This research advances previous work to create multi-disciplinary design tools that help designers generate innovative conceptual design solutions irrespective of their level of experience. This research builds on existing research including the Functional Basis, a finite set of terms used to define the desired or existing functionality of a product, a Component Naming Taxonomy that is based on a distinct function-based definition, a Design Repository that houses knowledge of existing consumer product designs, and a computational concept generator.

The current research develops and tests a system consisting of concept generation, product design and visualization tools (existing and new) with the goal of improving innovation and creativity in engineering design. To enable development of computational tools for product design using the Functional Basis, Component Naming Taxonomy, and Design Repository, the existing database schema of the Design Repository is modified to introduce new terms like the performance type, metric, etc., which are in turn used to define the concept of component interfaces (Input/Output). A modular product design software is then created based on the definition of Interface, which has additional functionality like working directly with the black box function model of the product, and inherently handling branching and merging. The flexibility of the algorithm is demonstrated with a case study. This modular product design software tool is developed using Integer Programming (IP) formulation to solve the design problem using Artificial Intelligence (AI) planning. Finally to enable better understanding of the output from the computational concept generator, the information about the terms in the Component Naming Taxonomy (also needed for the modular product design software) is collected and presented to the users in the form of open source wiki pages.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 0742693. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

Chapter 1

Introduction

The conceptual phase of design is a crucial step in the design of any product. The tools like brainstorming, morphological search, etc. that are used to help generate concepts (ideas), depend heavily on the experience and perception of the designers. Bryant, et al. [1] hypothesized that computational tools based on a repository of existing designs will free the designers from such constraints and help them quickly generate a lot more concepts. But the tool should not flood the user with a lot of irrelevant and incomplete ideas, and any user (independent of their experience and discipline) should be able to understand the results. Addressing some of these issues, a computational concept generator tool was developed by Bryant, et al. [1] that would help generate and evaluate conceptual designs.

The automated concept generator utilizes matrix-based computations applied on the product design information stored in an online Design Repository to create concepts. The most current information from the repository is captured as component functionality and compatibility information in the form of function component matrix (FCM) and design structure matrix (DSM). The program, developed in Java, takes as input a function adjacency matrix (FM), written by the user based on the functional model, that contains the desired design functionality. The algorithm then uses the information in the FCM and DSM to generate feasible ranked concepts for the given FM.

The output from the concept generator is displayed as a list of all the feasible components for each function term in the FM, which result in complete concepts. To reduce confusion due to the different naming of the components and to make the solutions more abstract, the recently revamped Component Naming Taxonomy is utilized [2]. The abstraction of

component terms allows users to be more creative in their solutions. To help users visualize and utilize the output from this tool to create innovative design solutions, the output is complemented by presenting information about the component terms like sample applications, general design considerations, images (of applications) and 3D CAD models using Solidworks and the open source Google SketchUp [3]. This information is collected and presented using user-editable, open source wiki pages [4] maintained by the innovation and design for engineering and systems (IDEaS) lab at the Pennsylvania State University.

Another approach to automate the design process for applications like product redesign and modular product design is explored in collaboration with Yoo, et al. [5]. The objective is to use the design information and component specifications from a repository of existing designs and components (by different manufacturers). This modular product design software, based on an integer programming (IP) formulation and solved using artificial intelligence (AI) planning [5], takes the black box model as input (instead of the functional model). This allows the modular product design software to search for more options as it is not restricted by the FM, but for the algorithm to work for these applications, some new fields, to define the artifacts (components), are introduced (e.g., parameter type, features, metric, Interfaces), which are based on the existing fields like flow. Based on the new fields, a modified database schema of the repository is proposed which allows this tool to make use of the information stored in the Design Repository. In an effort to expand the information scope of the repository, and to allow for easy communication of component information from across the globe, an XML-based module description language is introduced and used.

In the next chapter, existing research on automating the concept generation phase and the various computational tools that had been developed are studied as part of literature review. In chapter 3, the consolidated Component Naming Taxonomy is discussed. Also discussed is the information that is collected about each term of the Component Naming Taxonomy and how it is

presented in the form of wiki pages. In chapter 4, the definition of Interface is introduced and discussed in detail along with the supporting modified database schema. In chapter 5, the modular product design software is discussed in detail along with an application of the software to a product family of screwdrivers. In chapter 6, the contribution of this research in developing computational tools for conceptual design is presented along with discussion on future direction of work.

Chapter 2

Literature Review

In this chapter, the various design tools that are available to researchers for automating the conceptual design stage are reviewed. First, the Functional Basis is studied as a language to clearly define the functionality of a component. Second, the Component Naming Taxonomy to classify components based on functions, which is used extensively in this work, is studied. Third, the Design Repository and its underlying database schema are studied as a means for engineering designers to store existing design information. Finally a few of the computational tools, which are currently available to assist designers during concept generation and during product design, are studied.

2.1 Functional Basis

The Functional Basis [6-7] is a set of clearly defined design terminologies that helps product designers formulate and understand product functionality in a uniform way and is validated and popularly used by many researchers [8-10]. The Functional Basis requires users to specify verb-object (function-flow) pairs to define functions in a product. Due to its completeness, the entire functionality of components can be represented using the finite set of Functional Basis terms [6]. Functional Basis is used as a foundation for developing tools in many applications like biomimetic conceptual design [11], product platform based design [12], etc.

The hierarchical set of terms encompasses the boundary of language used in designing products in a limited set of terms, which facilitates computational software developments, such as automated concept generators, e.g., interactive morphological matrix computational design tool

(MEMIC) [13]; however, the very abstract nature of the defined function-flow terms makes it difficult to completely describe the uniqueness of each component as well as relationships between components, which is required in the cases of specific design like modular product design.

The terms in the functional set of the basis are divided into eight categories i.e., primary classes, which are further divided into secondary and tertiary levels, which provide further specialized description of the function. The primary and secondary function classes are listed in Figure 2-1. Also shown in the figure is the flow set of the basis which also has three hierarchical classes (the tertiary class is not shown). The primary class contains the popularly used classification of flow – Material, Signal and Energy [14]. The secondary class has 20 nouns that can define the input and output flows more clearly.

Functional Basis Reconciled Flow Set	Primary Class	Secondary Class
	Material	Human
		Gas
		Liquid
		Solid
		Plasma
		Mixture
	Signal	Status
		Control
	Energy	Human
		Acoustic
		Chemical
		Electrical
		Electromagnetic
		Hydraulic
		Magnetic
		Mechanical
		Pneumatic
		Radioactive/Nuclear
		Thermal

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

Figure 2-1. Primary and secondary function and flow terms of the Functional Basis [6]

2.2 Component Naming Taxonomy

A Component Naming Taxonomy references to a list of standard component terms recently reorganized by Kurtoglu, et al. [15] to abstractly identify the class of the artifact. The

component naming is based on a distinct function-based definition that is presented (the consolidated version) in Appendix A. Design tools like the Design Repository described in the following section and the automated concept generator [13] have implemented this taxonomy to classify product artifacts.

This type of classification enables another level of abstraction by encompassing similar components (applications) under one component name. For instance, the component taxonomy name Abrasive can mean – Sandpaper, Abrasive wheel, Diamond powder, or Pumice stone. By using the taxonomy in concept generation tools, the user will be given the output variant as Abrasive. The user can then lookup the applications of this term in the output visualization to decide what application s/he wants it to refer to, or for specific component names s/he can access the repository entries corresponding to the component taxonomy name. Using the naming scheme also eliminates redundancy and allows for well-defined and manageable (size) FCM and DSM data files. Components performing the same function(s) can have different names, for example components with names like Scanner Motor, Brush Motor, Lens Motor, Door Motor, etc all fall under the same Component Taxonomy Name – Electric Motor. Hence, using the Component Taxonomy Name further benefits tools like automated concept generator to give more unique and concept variant results.

2.3 Design Repository

The objective of a Design Repository is to provide a shared space for engineering designers to share their product design information. The database schema and Design Repository developed by Bohm, et al. [16] was utilized in this research. At the present time, the Design Repository is used not only as a hub for designers for information exchange but also as a design generation tool [16-18]. The key point of the Design Repository is to allow designers to store and

retrieve design knowledge at various levels of abstraction related to form, architecture, and function. To do this, the repository is divided into seven main categories of artifact-, function-, failure-, physical-, performance-, sensory-, and media-related information [16].

Practical application of the database schema (see Figure 2-2) from the Design Repository is essential in order to introduce the new fields to define Interface in Chapter 4. For that reason, detailed definition and usage for some categories such as artifact- and performance-design knowledge are essential. Artifact and performance-related design knowledge are core categories used in this research. The artifact table serves as a central hub for the various types of data. A product artifact hierarchy is created using a parent-child relationship between the data. Performance-related design knowledge also provides a high level embodied data schema to capture the overall functionality of various components.

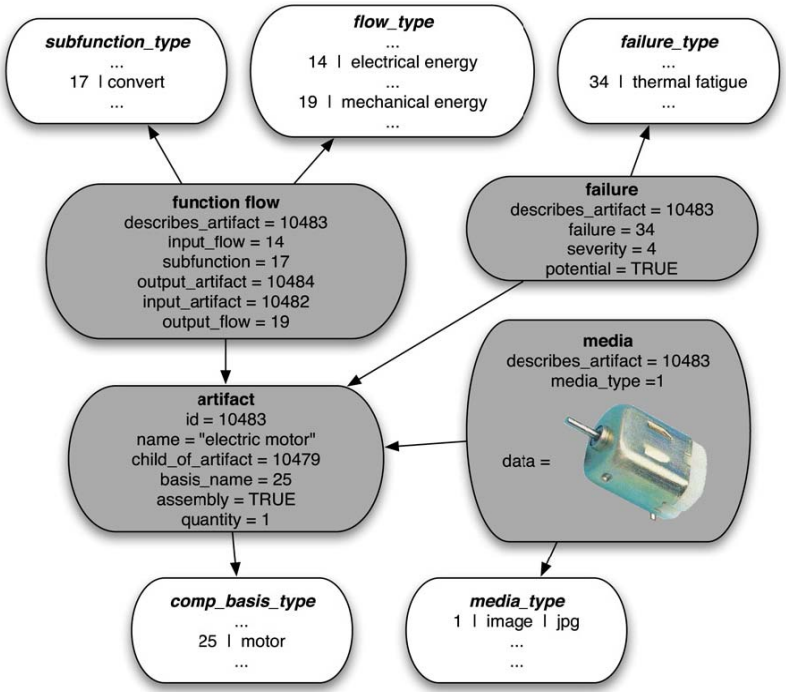

Figure 2-2. Repository Schema [16]

Emerging techniques in concept generation are one of the key design methods that utilize the Design Repository. Extracting and reusing component data to generate efficient conceptual solutions to new design problems is one of the main benefits of using the Design Repository. Originally developed and refined at the University of Missouri – Rolla and in collaboration with the University of Texas at Austin, Pennsylvania State University, Bucknell University, and Virginia Polytechnic Institute and State University [2], the Design Repository is currently hosted by the Design Engineering Lab at Oregon State University [19] and contains information on 146 products and 5710 artifacts (this number keeps increasing as new products are being added frequently). Any user can use the browse artifact option to look at the information of any component recorded in the repository as displayed in Figure 2-3.

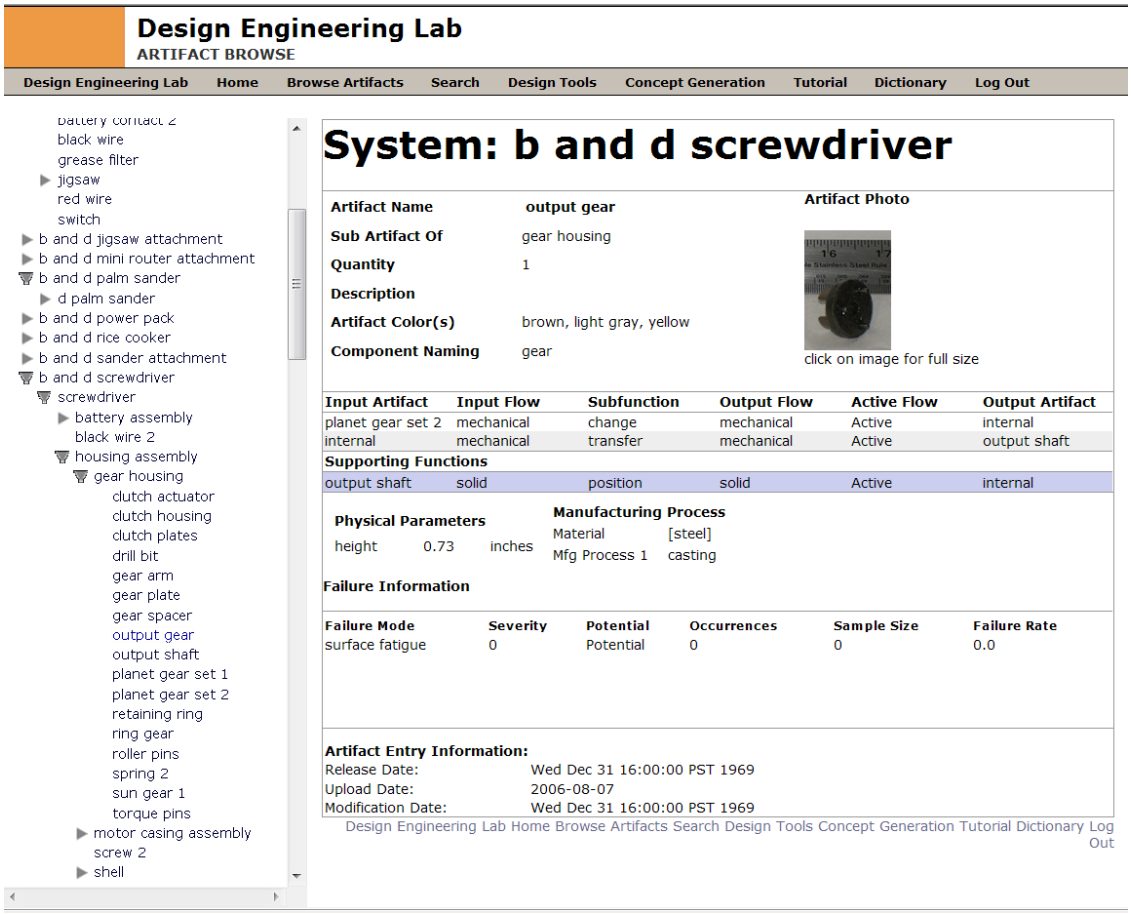

Figure 2-3. The browse page of the Design Repository [19]

2.4 Automated Concept Generator

Bryant, et al. [13] developed a computational tool to automate the concept design stage called automated concept generator (MEMIC). The goal of that research was to develop the tool for the early conceptual design phase based on the Functional Basis and information from the Design Repository in the form of the Function Component Matrix (FCM) and Design Structure Matrix (DSM). The first step by the designer is to construct a Functional Model from the Black Box Model, which is used as input to the tool in the form of the Function Adjacency Matrix (FM). The tool then lists components that satisfy each of the functions in the functional diagram with a corresponding ranking based on how often the component has been used for that function.

The automated concept generator quickly creates multiple concepts using the design knowledge stored in a repository, which eliminates the dependence on the experience of a team of designers. A single designer can easily use this tool to come up with multiple feasible concepts out of which the most innovative idea can be selected. Although concepts can be evaluated using parameter information (e.g., weight, cost) for each component, detailed properties of each module, such as voltage of electrical Interface, torque of a motor, etc. are not considered. More significantly, there are some major limitations in this approach when targeting different design phases. First, the tool does not easily handle function- and component-sharing when generating solution, which are key issues with modular designs. In addition, the tool cannot directly handle modules (components) with multiple inputs or outputs during concept generation. Also, the output is in the form of a list of component terms but without readily available information about those terms. Finally, the concepts generated by this tool are limited by how the user creates the Functional Diagram from the black box model.

2.5 Summary

The various tools available to automate the conceptual design (high level design) are studied in detail so that some of these can be modified or built upon to be able to be used to automate other design levels like automation of product design (low level design). Also studied are some of the limitations of the automated concept generator like the need for better information on the Component Taxonomy Terms which is addressed in the next Chapter.

Chapter 3

Consolidated Component Naming Taxonomy

The original Component Naming Taxonomy [2] is reorganized and consolidated by looking at the data from the Design Repository. Some of the component basis terms that have a very low frequency of occurrence in the Design Repository are combined with other terms (sometimes renamed as new terms) so that the consolidated classification is still comprehensive. The component terms and component subset terms are also consolidated together. The final consolidated classification contains 127 component terms, and they are all listed along with their function-based definition in Appendix A. The nature of this reduction in the number of component terms results in the reduction of the size of the FCM and DSM.

To facilitate designers to easily access information about the component taxonomy terms, their applications and to provide visual information to the users, open source wiki pages [4] are created and maintained by the research team in IDEaS Lab at Pennsylvania State University.

3.1 Component Naming Taxonomy Wiki Pages

The main Component Naming Taxonomy wiki page lists links to individual wiki pages of the terms from the consolidated Component Naming Taxonomy both as an alphabetized list and as a hierarchical list as shown in Figure 3-1. The individual pages for all the 127 component basis terms contain detailed information like definition of the term, sample applications, images of applications, 3D CAD Models – in both SolidWorks and Google Sketchup, general design considerations, relevant engineering concepts/equations, similar concepts (internal links), and external references.

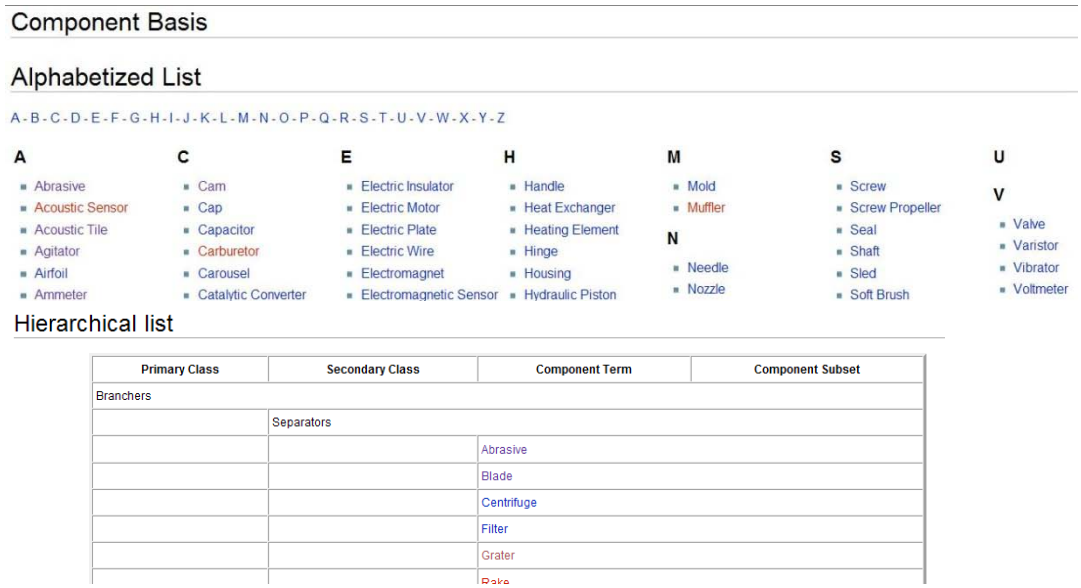

Figure 3-1. The main component basis wiki page

The function-based definition of each component basis term is provided from the component taxonomy definition table provided in Appendix A. The sample applications are collected from information available on the internet like Wikipedia pages and free online academic sites and the source is cited and links are provided in the External references section. In order to cover most of the possible applications, the artifacts available under each component basis term on the Design Repository are also examined. The flow terms associated with each of the application are also recorded. This information is used in Chapter 4, during the definition of Interface, to decide on which Interfaces (defined on flow terms) to use for each of the components. In an effort to provide visual information and as means for the user to visually interact with the concept variants from the concept generator, images and 3D CAD models are created by Rao [20] and provided on the wiki page. Excerpts from a component basis term (abrasive) wiki page are provided in Figure 3-2.

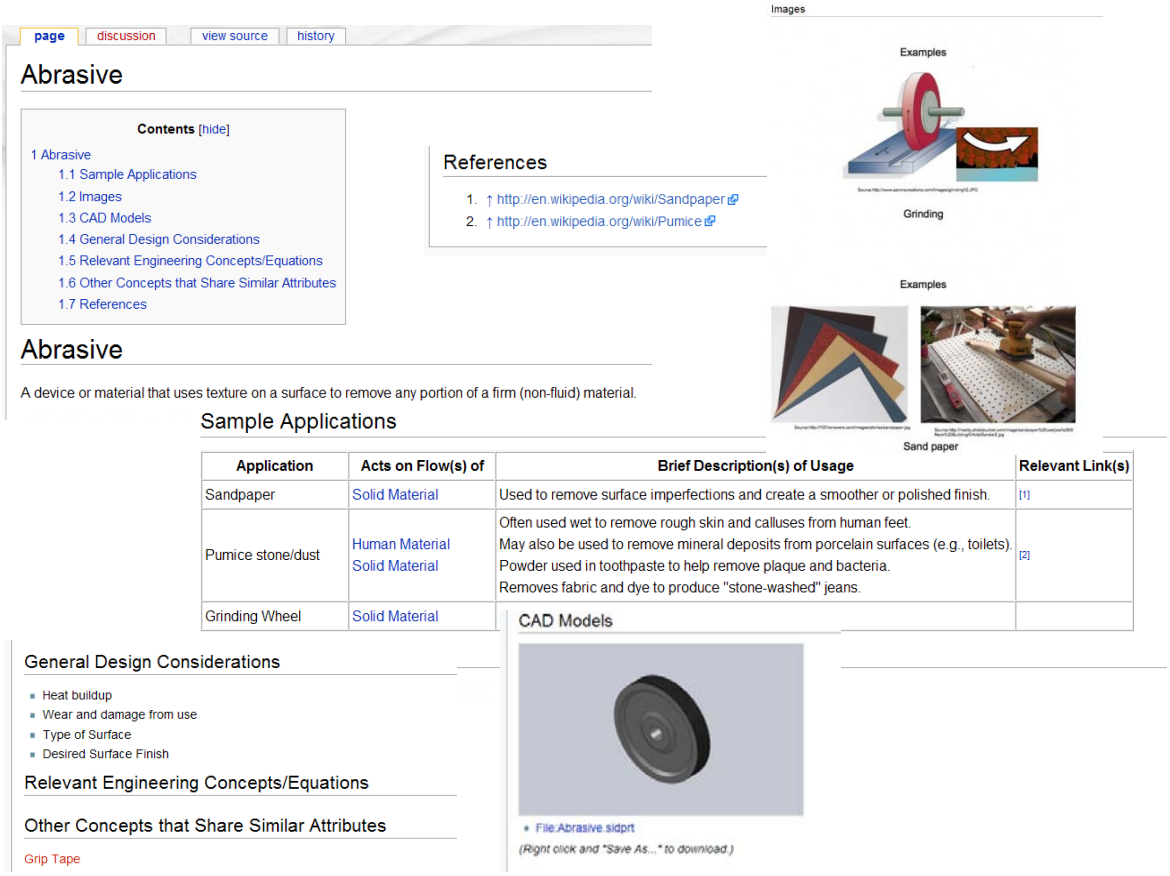

Figure 3-2. Excerpts from the abrasive wiki page

3.2 Enhancing the Concept Generator Output

The latest version of the automated concept generator – MEMIC software [13], is a Java based application that requires three text files as input in order to generate concepts. The first file is the matrix form of the Functional Model (created from the black box model by the user) called the Function Adjacency matrix (FM) saved as a tab delimited text file. The other two files are the FCM and DSM that are generated from the Design Repository [19] using its Design Tools. The component basis naming of the artifacts is used while generating these tools. The input screen when the MEMIC is started looks like in Figure 3-3.

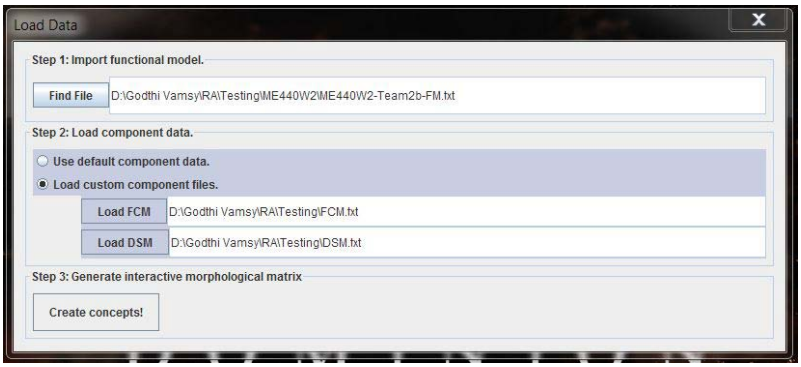

Figure 3-3. MEMIC input

After the files are loaded and 'Create Concepts!' is clicked, the tool output shown in Figure 3-4 will show all the functions present in the input FM. The drop-down menu adjacent to the function lists all the feasible component terms that can make a complete concept for the input FM. The percentage displayed in parenthesis gives the rank of the component selected, based on the frequency that the component is used to solve that particular solution. The ranking statistics window in Figure 3-4 can be accessed from the window menu and displays the current selected (components) concept on a scale of Least common to Most common configuration. The designer can then access information about the component terms to get a better understanding of the output and to decide which solution to choose.

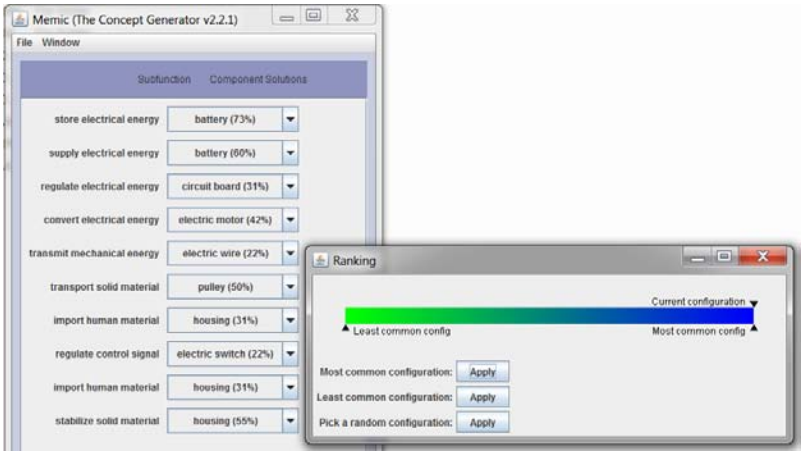

Figure 3-4. MEMIC output and ranking window

3.3 Summary

The Component Naming Taxonomy is used as another level of abstraction in design and the advantage of using this is discussed. Since some of the design tools like the concept generator use the terms of the Component Naming Taxonomy as the output, various information about each term like definition, sample applications with flow information, images, and 3D CAD Models, general design considerations, etc. are provided to designers through open source wiki pages. The potential to use the flow information for the component terms in conjunction with the interface definition to specify a component/module is also introduced. The next chapter introduces the Interface definition.

Chapter 4

Interface

From the literature review of the existing work it was observed that for computational tools like the concept generatior [6], the components are described by the functions that they satisfy and the input and output flows associated with them. The component information is then stored in a Design Repository [5]. The way the flows are currently defined makes it difficult to distinguish between different types[1] of similar components/modules, but this distinction is required for modular product design/redesign. Also the Design Repository associates one input and output flow with each function, and if one component satisfies two or more functions, then they are listed separately with one input and output per function. Due to these constraints, the current schema is limited in handling input/output flow descriptions of components.

To address the issue of clearly defining the input and output flow of components, it is proposed to define input/output Interfaces of components stored in the Design Repository. One way of defining this is to use specific names for each Interface between components/modules as in [13], but this hinders the design search tool by limiting the possible combinations it can search and it is not always possible to have specific names for Interfaces. Another way is to use the flow information itself but by providing more details about the flow. This is the approach that is followed in defining the Interface in the next section. Section 4.2 defines the parameters and features. Section 4.3 proposes a modified database schema to be implemented in the Design Repository to support the Interface information.

[1] Say for instance a component that **converts electrical energy to mechanical energy** is associated with a motor in the existing scenario but it can be a 6 V DC motor, 12 V DC motor, 110 V AC motor, 220 V AC motor. Based on the various global markets and application, a different type may need to be selected. Also the torque requirement can vary based on application.

4.1 Definition of Interfaces

After examination of many components in the repository, the definition of an input or output Interface is proposed as the corresponding input or output flow(s) for a component/module[2] which are defined by the (i) performance type, (ii) metric, and (iii) range of the value of the metric(s). The definition of terms used here is based on the repository database tables [16] and Functional Basis [13], but they have been modified and added as necessary. The performance type is a way of specifying the flow. The metric is the unit of measurement used for the performance type. The range defines the various values (or range of values) of the performance type (in the metric) that the component/module can handle. For example, the flow "electrical energy" can have the performance type "Voltage" with the metric as "Volt" and ranges of (6 – 7) or (110 – 120), etc; while "mechanical rotational energy" can have the performance types "Torque" and "Speed" with corresponding metrics "inch-pound" and "rpm".

The performance types can be identified for all the flows by using definitions of each of the flows as provided in the Functional Basis . A common metric for each performance type is to be identified[3] and used. For the flow terms in the Functional Basis, the metrics are identified as shown in Table 4-1. The range values keep getting added into the database as component details are entered so that when a new component is being entered or when a query is being made, the value of ranges is controlled within the pre-defined data entry.

In Table 4-1, it can be observed that the performance types and the metrics for the flows under the primary category of Material and Energy are determined by looking at some of the measurement standards and also by the applications (context) in which these flow terms are used. Different applications merit different performance types. For example, the rotational energy,

[2] Where a module can be a subassembly or just a cluster of components

[3] Thinking in a global perspective, each country follows a particular standard like the SI system, IPS system, etc, but they can all be converted internally during data entry into the standard used by the repository.

when used in a braking (no speed) application, can be specified solely by torque, but, when used in a rotating energy transfer application, the flow metrics need to include both torque and speed. By observing the applications of all the component terms and their associated flows, Table 4-1 is completed. The flows such as those categorized under signal are even more dependent on the application, and hence the list provided in Table 4-1 is not a comprehensive one.

Table 4-1. Interface definition of all the flow terms <table continued on page 19>

Flow Terms			**Performance Type**		**Metric**	
Primary	**Secondary**	**Tertiary**	**1**	**2**	**1**	**2**
Material	Human		Mass-Human		Kg	
	Gas		Actual Volume-Gas		m^3	
	Liquid		Volume-Liquid		m^3	
	Solid	Object	Mass-Solid		Kg	
		Particulate	Mass-Solid		Kg	
		Composite	Mass-Solid		Kg	
	Plasma		Volume-Plasma		m^3	
	Mixture	Gas-gas	Actual Volume-Gas-Gas		m^3	
		Liquid-Liquid	Volume-Liquid-Liquid		m^3	
		Solid-Solid	Mass-Solid-Solid		m^3	
		Solid-Liquid	Volume-Solid-Liquid		m^3	
		Liquid-Gas	Volume-Liquid-Gas		m^3	
		Solid-Gas	Volume-Solid-Gas		m^3	
		Solid-Liquid-Gas	Volume-Solid-Liquid-Gas		m^3	
		Colloidal	Volume-Colloidal		m^3	
Signal	Status	Auditory	Sound Pressure Level	RMS Sound Pressure	dB	Pa
		Olfactory	Electric Potential		volt (V)	
		Tactile	Electric Potential	Current	volt (V)	ampere (A)

		Taste	Electric Potential Difference[4]		volt (V)	
		Visual	Luminous Intensity		Candela (cd)	
	Control	Analog	Electric Potential		volt (V)	
		Discrete	Electric Potential		volt (V)	
Energy	Human		Force		N	
	Acoustic		Sound Pressure Level	RMS Sound Pressure	dB	Pa
	Biological		Energy-Biological		kJ	
	Chemical		Molar Chemical Energy		kJ/mol	
	Electrical		Electric Potential & Current	Electrical Energy Consumed	volt (V) & ampere (A)	kWh
	Electromagnetic	Optical	Planks Constant & Frequency	Luminous Intensity	Candela (cd)	J-s and Hz
		Solar	Planks Constant & Frequency		J-s and Hz	
	Hydraulic		Pressure-Hydraulic & Area & Velocity	Pressure & Area	Pascal (Pa) & m^2 & m/s	Pascal (Pa) & m^2
	Magnetic		Magnetic Flux	Magnetic Flux Density	weber (Wb)	tesla (T) (N/A-m)
	Mechanical	Rotational	Torque & Rotation Speed	Torque	N-m & r/s	N-m
		Translational	Force & Translation Speed	Force	Newton (N) & m/s	Newton (N)
	Pneumatic		Pressure-Pneumatic & Area & Velocity	Pressure & Area	Pascal (Pa) & m^2 & m/s	Pascal (Pa) & m^2
	Radioactive/Nuclear		Radiation Energy	Binding Energy	kiloelectron volt (keV)	Megaelectron volt (MeV)
	Thermal		Thermal Energy		Joule (J)	

4.2 Definition of Parameters and Features

The details about the components, other than the flow information, like the cost (material, manufacturing, etc.), weight, and dimensions (length, width, height, diameter, etc.) are defined in the parameter type. Just like the Interface, the parameter type also has a metric and a range (or

[4] The taste sensors (in an electronic tongue) receive information about five categories - sourness, saltiness, bitterness, sweetness, and umami (deliciousness) – by comparing with a reference for each and **outputs electric potential difference** corresponding to the variation from reference.

single value) associated with it [5], but to specify a component fully there might be some other terms needed that can neither be defined as input/output Interfaces nor can they be defined as parameters. To fully specify some components (for specific applications) the information other than interface and parameter types that needs to be associated with the component is defined as feature type.

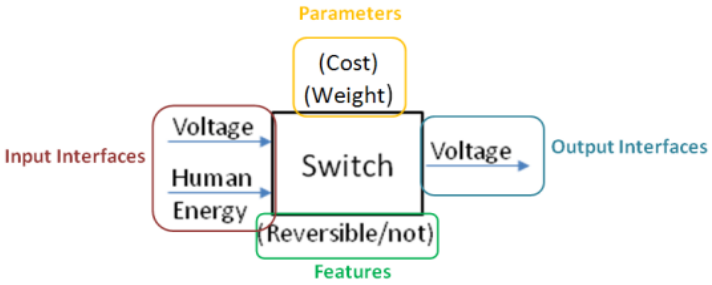

Figure 4-1. The complete specification of a component (switch)

Consider for example, the component term “Switch” shown in Figure 4-1which can have “Human Energy and Voltage” as input Interfaces and “Voltage” as an output Interface, but the switch can also have the ability to reverse its polarity. So, to completely define the Switch[5], this information needs to be included. Since this is not an interface or a parameter, it is defined as a feature. Since the definition of Interfaces refers to the flow definitions of the Functional Basis, the terms are controlled and pre-defined; however, the definition of features does not yet have a defined set of terms, which needs to be addressed in future work to ensure consistent definitions across components.

By using the information about the flow associated with each term in the Component Naming Taxonomy (provided on the wiki pages), the input/output interfaces of each of the component terms, for different applications, can be decided. The ability of the Interface definition to completely define some component terms is shown in Figure 4-2. For these components,

[5] The price would be different from the one not having such feature and since such a feature changes the objective of minimizing cost, it needs to be considered while specifying the switch. Also depending on the design requirements such a feature may need to be included or excluded.

sample artifact data - relating to the input/output Interfaces, parameters and features - is also collected and tabulated in Appendix B.

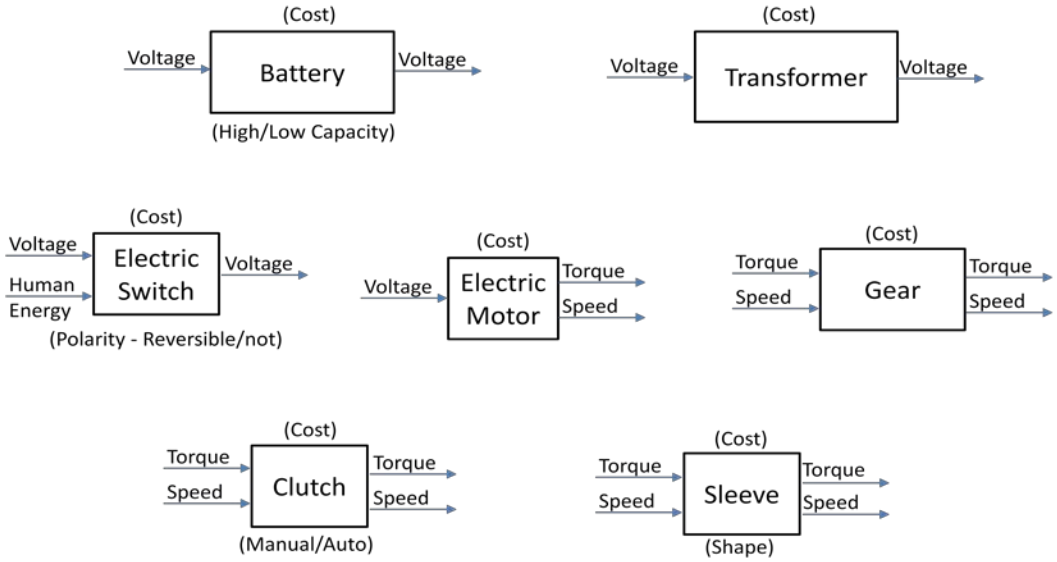

Figure 4-2. The complete specification of some other components

4.3 Supporting Database Schema

A database schema is needed to capture and store all of the Interface information. To maintain consistency and to be able to integrate the work into the existing Design Repository, the database schema shown in Figure 4-3 was developed based on the current Design Repository data schema [16]. It follows the same structure and uses the same nomenclature as the existing database schema but adapted to the purpose of storing Interface information. For instance, the Performance Characteristics table has been modified to include two new fields “Input/Output” and “Range”. The Feature table and its associated supporting tables and the Range table are newly introduced to enhance the scheme to enable the aforementioned Interface definitions.

Main Tables

Artifact Table		
Field	**Data Type**	**Default**
Id	serial	
Name	Varchar	
Assembly?	Boolean	FALSE
System	Int	
Quantity	Int	1
Manufacturer/Supplier	Int	

Function Flow Table		
Field	**Data Type**	**Default**
Id	Serial	
Describes_Artifact	Int	
Supporting	Boolean	FALSE
Input_Flow	Int	
Subfunction	Int	
Output_Flow	Int	

Parameter Table		
Field	**Data Type**	**Default**
Id	Serial	
Describes_Artifact	Int	
Parameter_Type	Int	
Parameter_Metric_Type	Int	
Parameter_Value	Float	

Performance Characteristics Table		
Field	**Data Type**	**Default**
Id	Serial	
Describes_Artifact	Int	
Performance_Type	Int	
Performance_Metric_Type	Int	
Performance_Value	Float	
Input/Output	Boolean	
Range	Int	

Feature Table		
Field	**Data Type**	**Default**
Id	Serial	
Describes_Artifact	Int	
Feature_Type	Int	
Feature_Metric_Type	Int	
Feature_Value	Boolean	

Manufacturer/Supplier Info Table

Supporting Tables

System Table		
Field	**Data Type**	**Default**
Id	Serial	
System_Name	Varchar	
System_Type	Int	

Subfunction Type Table		
Field	**Data Type**	**Default**
Id	Serial	
Subfunction_Type	Varchar	
Tier	Int	

Parameter Metric Type Table		
Field	**Data Type**	**Default**
Id	Serial	
Parameter_Metric_Name	Varchar	

Performance Metric Type Table		
Field	**Data Type**	**Default**
Id	Serial	
Performance_Metric_Name	Varchar	

Parameter Type Table		
Field	**Data Type**	**Default**
Id	Serial	
Parameter_Name	Varchar	

Performance Type Table		
Field	**Data Type**	**Default**
Id	Serial	
Performance_Name	Varchar	

System Type Table		
Field	**Data Type**	**Default**
Id	Serial	
System _Type_Name	Varchar	

Flow Type Table		
Field	**Data Type**	**Default**
Id	Serial	
Flow_Type	Varchar	

Range Table		
Field	**Data Type**	**Default**
Id	Serial	
Performance_Type	Int	
Range	Varchar	

Feature Type Table		
Field	**Data Type**	**Default**
Id	Serial	
Parameter_Name	Varchar	

Feature Metric Type Table		
Field	**Data Type**	**Default**
Id	Serial	
Performance_Name	Varchar	

Figure 4-3. Proposed modified database schema

4.4 Summary

The input/output Interface is defined based on the flow terms in the functional basis [8] and recorded in Table 4-1. The ability to make use of the flow information corresponding to different applications of the Component Naming Taxonomy terms to specify interfaces is observed by the complete specification of some component terms. This sets the stage for an Interface-based machine readable language that can be used in computational tools to support product design like the modular product design software introduced in the next chapter.

Chapter 5

Modular Product Design Software

Modularized components produced by suppliers from around the world can shorten product development cycles and reduce costs by using similar modules across multiple products. Rapidly changing customer demands also dictate that manufacturers shorten their product development cycles in order to remain competitive. To achieve this, it is essential to address the following three challenges [5]: (1) develop an Interface-oriented machine-readable representation scheme for modular components; (2) formalize an accessible cyber-infrastructure-based framework that enables global users to describe, publish, and discover components information in a standardized way; and (3) develop automated modular product design software. To address these challenges, the modular product design software is developed jointly with Yoo in [21].

Figure 5-1 illustrates the overall architecture of the proposed cyber-infrastructure for global manufacturing, where suppliers can upload their component information in a machine-readable format in a real-time basis into a global Design Repository regardless of their geographical locations. This global repository is based on the proposed modified database schema to capture Interface specific information. In addition, a standardized representation scheme utilizing XML enables suppliers to describe their components in a machine-readable way for easy communication between suppliers, the repository and the software.

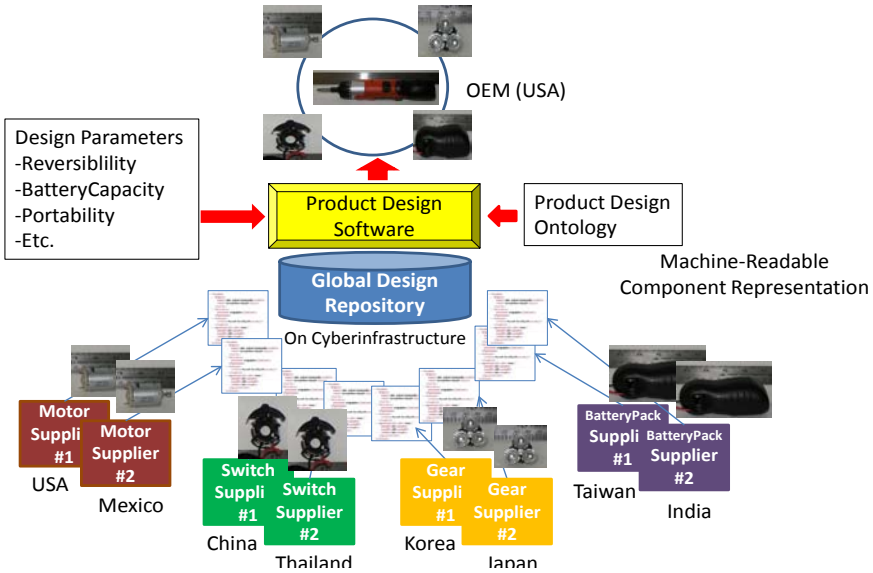

Figure 5-1. Modular product design through global collaboration [5]

5.1 Module Description Language (MDL)

Chapter 4 defines the proposed Interfaces for components, which can be extended to modules. This section proposes a formal way of describing modules based on the Interface definition, which is useful to standardize the product design processes. eXtensible Markup Language (XML) is used to represent modules, which is widely used in the field of computer science [22]. The main characteristic of XML is its machine-readability. Elements and attributes are two main components of XML. For example, in Figure 5-2, “geometricSpecification” is an element, and “width” is an attribute of the element. Once the format of XML is defined in XML Schema or DTD (Document Type Declaration), all instances of XML are verified syntactically through a validity check so that syntactic errors can be minimized. After the syntax verification, the content of an XML file can be read using various XML parsing software libraries, such as SAX (Simple APIs for XML) or DOM (Document Object Model).

The Module Description Language (MDL) introduced in Figure 5-2 is based on this XML schema [23]. MDL adopts the concept of Interface-oriented module description, as proposed in Chapter 4. The first-level element is “Component,” from which it is clear that each MDL describes one component at a time. The “Component” element has four sub-elements, “generalInfo”, “inputs”, “outputs”, and “features”. The contents of the “generalInfo” element include each piece of information of a company name, a component name, price, weight, and size of the component. The contents of the “inputs”, “outputs”, and “features” elements include information of input Interfaces, output Interfaces, and features of the component.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <Component xmlns:tns="http://product.repository"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://product.repository Component.xsd">
  - <generalInfo>
      <companyName>BlackAndDecker</companyName>
      <componentName>Switch</componentName>
      <price>0.14</price>
      <weight>0.1</weight>
      <geometricSpecification width="1.5" height="1.0" depth="0.5" />
    </generalInfo>
  - <inputs>
      <input performanceType="Voltage" performanceMetric="Volt" performanceValue="1" />
      <input performanceType="HumanEnergy" performanceMetric="Joules" performanceValue="2" />
    </inputs>
  - <outputs>
      <output performanceType="Voltage" performanceMetric="Volt" performanceValue="1" />
      <output performanceType="Radio" performanceMetric="Frequency" performanceValue="21" />
    </outputs>
  - <features>
      <feature featureType="Reversibility" featureValue="Yes" />
    </features>
</Component>
```

Figure 5-2. The proposed Module Description Language (MDL)

The proposed MDL would be provided by suppliers for their modules (and components) and stored in a global Design Repository, which can be administrated by an individual manufacturer or an international oversight organization. In this manner, the Design Repository would remain up-to-date internationally since suppliers from all over the world can upload their newly-introducing module (and component) information in a real-time basis.

5.2 Integer Programming Formulation

This modular product design problem is formulated using Integer Programming (IP), based on the work by Yoo, et al. [5], in which the problem is transformed into an Artificial Intelligence (AI) planning problem. The IP formulation can incorporate not only functional requirements, such as compatibility, but also non-functional requirements, such as cost, weight, or geometric constraints. Especially, non-functional requirements are handled exceptionally well in the IP approach. The formulation includes an objective function and constraints. The objective function can be any numerically definable expression. For example, if the target market segment is a student or low-income group, the minimization of price can be the objective. Furthermore, multiple objectives can be also defined in this IP formulation. Initial constraints are given as Interfaces or features, such as 110 voltage power supply or human force. Goal constraints describe the goal features or Interfaces. The IP formulation also checks compatibility among modules. In other words, input and output Interfaces of a module must be correctly connected to adjacent modules. The variables and the equations are discussed in detail in Appendix C.

5.3 System Architecture

Figure 5-3 illustrates the architecture of the proposed cyber-infrastructure as well as how the system works. The cyber-infrastructure consists of the global Design Repository, modular product design service providers, and users. In most cases, the users are the manufacturers. The Design Repository can be run by individual manufacturers or an international oversight organization. The modular product design services can be provided by third-party service providers or manufacturers.

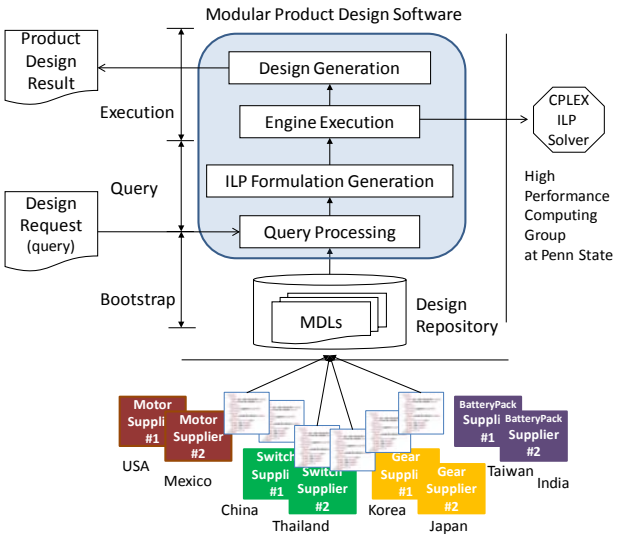

Figure 5-3. Detailed system architecture for product design

The modular product design software consists of four sub-modules: (1) query processing, (2) Integer Linear Programming (ILP) formulation generation, (3) Engine execution, and (4) design generation. First, the query processing module parses design requests written in XML as shown in Figure 5-4, which is generated based on the black box model, shown in Figure 5-5. The XML type design request includes initial conditions, which are energy sources in most cases, and target performance conditions or desired features, such as required output torque or switch reversibility, respectively. Second, the ILP formulation generation module generates ILP formulations based on the design request written in the XML type query and the MDL representation of each component stored in the repository. Third, after generating ILP formulations, the execution engine feeds the formulation to CPLEX [24], an Integer Programming solver. Fourth, the design generation module receives the solution result from CPLEX, interprets it, and finally generates a feasible product design. If there are no feasible designs, then the module returns a message of infeasible solutions. Figure 5-6 shows the CPLEX software finding an optimal solution to a design request in terms of cost and displays the list of components and their sequence.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Query xmlns:tns="http://product.repository" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://product.repository Component.xsd">
- <initialPerformanceInstances>
    <initialPerformanceInstance performanceType="Voltage" performanceMetric="Volt" performanceValue="8" />
    <initialPerformanceInstance performanceType="HumanEnergy" performanceMetric="Joules" performanceValue="2" />
  </initialPerformanceInstances>
- <goalPerformanceInstances>
    <goalPerformanceInstance performanceType="Torque" performanceMetric="InPerLb" performanceValue="5" />
    <goalPerformanceInstance performanceType="Speed" performanceMetric="RPM" performanceValue="6" />
    <goalPerformanceInstance performanceType="HoldsBit" performanceMetric="InterfaceType"
      performanceValue="7" />
  </goalPerformanceInstances>
- <goalFeatureInstances>
    <goalFeatureInstance featureType="Reversibility" featureValue="Yes" />
    <goalFeatureInstance featureType="HighLow" featureValue="Yes" />
    <goalFeatureInstance featureType="ModeselectNot" featureValue="Yes" />
  </goalFeatureInstances>
</Query>
```

Figure 5-4. Design request (query) in XML

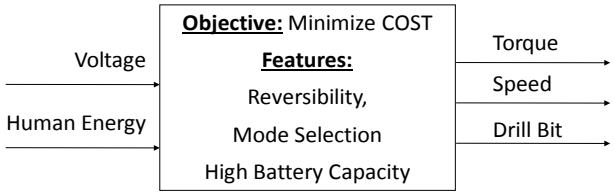

Figure 5-5. Black box model (Query)

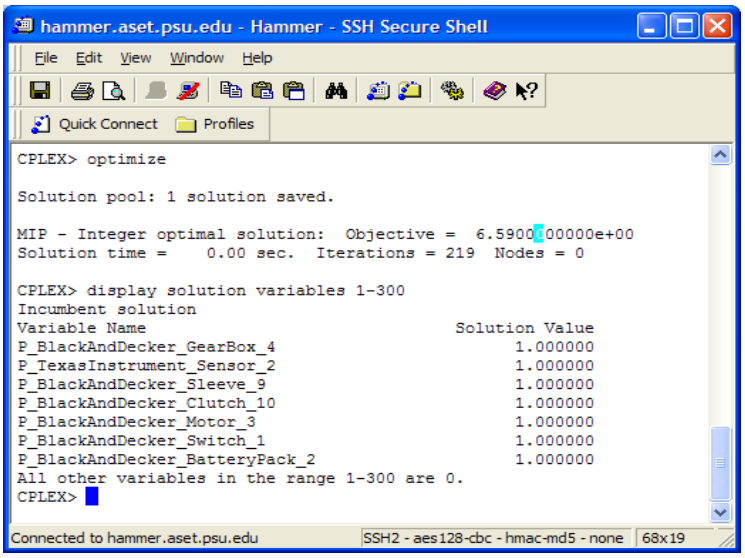

Figure 5-6. A snapshot of the CPLEX results

5.4 Application – Product Family of Screwdrivers

In order to show a potential application of the modular product design software, a hypothetical market segmentation grid [25] for Black & Decker electronic screwdrivers is created to illustrate how to create a product family of screwdrivers using our software tool. The market is divided into four different segments based on price and power supply type: A = portable luxury, B = home luxury, C = portable standard, and D = home standard, as shown in Figure 5-7. Luxury products have two additional features – reversibility and mode selection – as compared to the standard products. By offering reversibility, the screwdriver provides not only clockwise but also counter-clockwise rotation. The mode selection feature enables the electric screwdriver to be operated manually when needed. Availability of these features is captured in the Feature table of the artifact data schema. The difference between home and portable type is power supply. The portable model uses batteries to expand the work area; the home model uses the rated power supply (110V) as the power source for the product.

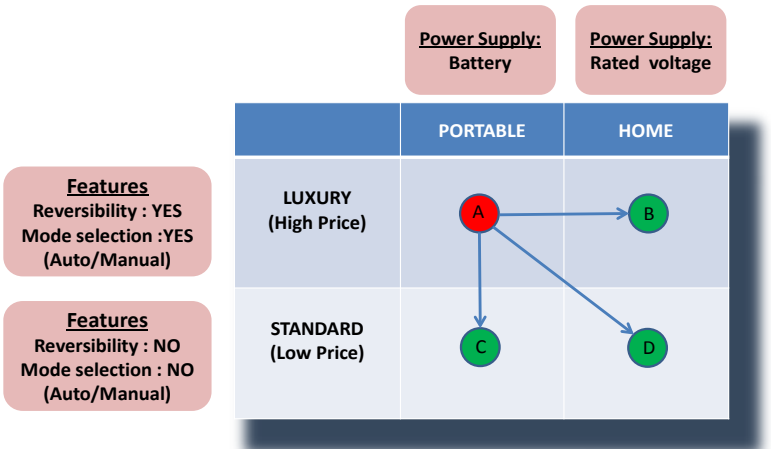

Figure 5-7. Market segmentation of electronic screwdrivers

Table 5-1 shows the initial and goal states for the portable luxury model as an example. As can be seen in Table 5-2, the best result for each component (artifact) is selected by the modular product design software based on the initial and goal states of the portable luxury model. For this application, 43 artifacts were created, including six components from existing Black & Decker screwdrivers, which were dissected as part of this study. The whole component (artifact) data is provided in Appendix B. In Table 5-2, there are three different sections: (1) input, (2) output, and (3) feature. Each section contains the information that represents the characteristic of individual component. The detail description of each Range is summarized in Table 5-3.

It is possible to describe the detailed information of each artifact by matching the ID value of the Range table with the actual range. For example, the electronic motor is one of the artifacts in a screwdriver. In Table 5-2, the motor depicts input with volt type and number is 1, first output with torque type and number is 3, and second output with speed type and number is 4. The actual meaning of each number is determined from the Range table in Table 5-3. Therefore, the electronic motor in the Black & Decker screwdriver can be represented by the artifact that transforms the electronic energy (range from 6 to7V) to rotational energy (torque range from 2 to 3 In/Lb and speed from 100 to 200 RPM).

Table 5-1. States for the portable luxury model

Initial		**Goal**	
Volt	1	Torque	5
Human Energy	2	Speed	6
		Shape	7
		High/Low	yes
		Reversibility	yes
		Mode select/Not	yes

Table 5-2. Result for the portable luxury model

Part	Company	Price	Weight	Input				Output				Feature	
Battery Pack	B&D	0.72	0.9	Volt:	1			Volt:	1			High/Low:	yes
Switch	B&D	0.14	0.1	Volt:	1	HE:	2	Volt:	1			Reversibility:	yes
Motor	B&D	2.20	0.5	Volt:	1			Torque:	3	Speed:	4		
Gear box	B&D	1.02	0.4	Torque:	3	Speed:	4	Torque:	5	Speed:	6		
Clutch	B&D	0.70	0.2	Torque:	5	Speed:	6	Torque:	5	Speed:	6	Mode select/Not:	yes
Sleeve	B&D	0.31	0.3	Torque:	5	Speed:	6	Torque:	5	Speed:	6	Shape:	7
Sum		5.09	2.40										

Table 5-3. Description of range table

ID	Performance Type	Range	Metric
1	Volts	6 ~ 7	V
2	Human Energy (HE)	10 ~ 20	Joules
3	Torque	2 ~ 3	In/Lb
4	Speed	100 ~ 200	RPM
5	Torque	15 ~ 30	In/Lb
6	Speed	30 ~ 60	RPM
7	Shape	Hex	
8	Volts	100 ~ 120	V
9	Volts	210 ~ 230	V
10	Torque	4 ~ 5	In/Lb
11	Speed	70 ~ 90	RPM
12	Shape	Square	

Table 5-4 shows the initial and goal status for the portable standard model, and Table 5-5 presents the optimal solution for these settings. Note that all values in the Feature section are “no” in the optimal solution for portable standard model (see Table 5-5). In addition, other input and output range values are matched component by component. Total price for this optimal solution is $4.76. It is cheaper than portable luxury model which is $5.09.

Table 5-4. States for portable standard model

Initial		**Goal**	
Volt	1	Torque	5
Human Energy	2	Speed	6
		Shape	7
		High/Low	no
		Reversibility	no
		Mode select/Not	no

Table 5-5. Result for portable standard model

Part	**Company**	**Price**	**Weight**	**Input**				**Output**				**Feature**	
Battery Pack	B-1	0.62	0.50	Volt:	1			Volt:	1			High/Low:	no
Switch	S-1	0.11	0.02	Volt:	1	HE:	2	Volt:	1			Reversibility:	no
Motor	B&D	2.2	0.5	Volt:	1			Torque:	3	Speed:	4		
Gear box	B&D	1.02	0.4	Torque:	3	Speed:	4	Torque:	5	Speed:	6		
Clutch	C-1	0.50	0.01	Torque:	5	Speed:	6	Torque:	5	Speed:	6	Mode select/Not:	no
Sleeve	B&D	0.31	0.3	Torque:	5	Speed:	6	Torque:	5	Speed:	6	Shape:	7
Sum		4.76	1.73										

All values in the feature section are “no” in the optimal solution for portable standard model (Table 5-5). In addition, other input and output range values are matched component by component. Total price for this optimal solution is $4.76. It is cheaper than portable luxury model which is $5.09.

Similarly it is seen that the optimal solution fully satisfied all initial and goal states for the home standard model (Table 5-6 and Table 5-7). To achieve these conditions, the voltage convertor is added in the optimal solution. The total price is $5.16. It is higher than the portable standard model because of the voltage convertor.

Table 5-6. States for home standard model

Initial		Goal	
Volt	8	Torque	5
Human Energy	2	Speed	6
		Shape	7
		High/Low	no
		Reversibility	no
		Mode select/Not	no

Table 5-7. Result for home standard model

Part	Company	Price	Weight	Input				Output				Feature	
Battery Pack	B-3	0.24	0.95	Volt:	8			Volt:	8			High/Low:	no
Voltage Convertor	V-1	0.78	0.84	Volt:	8			Volt:	1				
Switch	S-1	0.11	0.02	Volt:	1	HE:	2	Volt:	1			Reversibility:	no
Motor	B&D	2.20	0.50	Volt:	1			Torque:	3	Speed:	4		
Gear box	B&D	1.02	0.40	Torque	3	Speed	4	Torque:	5	Speed:	6		
Clutch	C-1	0.50	0.01	Torque	5	Speed	6	Torque:	5	Speed:	6	Mode select/Not:	no
Sleeve	B&D	0.31	0.30	Torque	5	Speed	6	Torque:	5	Speed:	6	Shape:	7
Sum		5.16	3.02										

The final product is the home luxury model as defined in Table 5-8. The optimal price is $6.00 as shown in Table 5-9. In this result, the voltage convertor is excluded from the solution. Instead of using a voltage convertor, the software suggests using the M-1 electronic motor in order to achieve the specified goal states based on the initial states (see Table 5-8).

Table 5-8. States for home luxury model

Initial		Goal	
Volt	8	Torque	5
Human Energy	2	Speed	6
		Shape	7
		High/Low	yes
		Reversibility	yes
		Mode select/Not	yes

Table 5-9. Result for home luxury model

Part	Company	Price	Weight	Input				Output				Feature	
Battery Pack	B-2	0.87	0.98	Volt:	8			Volt:	8			High/Low:	yes
Switch	S-3	0.61	0.22	Volt:	8	HE:	2	Volt:	8			Reversibility:	yes
Motor	M-1	1.50	0.11	Volt:	8			Torque	10	Speed	11		
Gear box	G-3	2.01	0.47	Torque:	10	Speed	11	Torque	5	Speed	6		
Clutch	B&D	0.7	0.2	Torque:	5	Speed	6	Torque	5	Speed	6	Mode select/Not:	yes
Sleeve	B&D	0.31	0.3	Torque:	5	Speed	6	Torque	5	Speed	6	Shape:	7
Sum		6.00	2.28										

5.5 Summary

The Module Description Language (MDL) is introduced as a means of communicating component specifications across different segments (manufacturers, global repository, designers). The proposed modular product design software is based on an integer programming (IP) formulation and solved using artificial intelligence (AI) planning and tested with an application (a product family of screwdrivers). The contributions, limitations and the future work of this research are discussed in the next chapter.

Chapter 6

Conclusions

6.1 Discussions – Modular Product Design Software

6.1.1 Contributions

Most computational tools for concept design like the concept generator developed by Bryant, et al. [6] use a multi-step process. First, using the design requirements, the overall system is represented by a black box model. Then the system architecture is captured in the form of a functional model. By replacing the functions with components performing that particular function, products are then created. While this is an effective way to create conceptual designs, there are limitations to this process: (1) the designer has to first build the system as a functional model and hence the solutions are limited by this, (2) the solution is thus limited to one-to-one mapping between the functions and components, (3) branching and merging in the functional model is not inherently handled but is split into separate linear chains [6]. Due to these limitations, the concept generator (high level design) falls short for applications like modular product (re)design and design of new products (low level design) where the goal is to reuse existing components in the database to create a product satisfying required features and constraints like cost, volume, weight, etc.

The proposed modular product design software addresses these issues successfully. The design requirements are provided as a black box model to the software. Thus, by eliminating the need for the intermediate step of manually creating the functional model, the advantages are (1) set-up time is reduced and (2) the software can explore all feasible combinations of components

to arrive at an optimal solution. Further, by defining the black box model based on its Interfaces, a new ability is created that was not possible using only flow descriptions without detailed parameters as in the previous tool. By skipping the step of using the functional models, the issue of one-to-one mapping between the functions and components does not arise, and the proposed software can deal with multiple inputs and outputs to each component. The approach by which the software searches the component space to create feasible solutions makes the software capable of handling branching and merging as well. Figure 6-1 illustrates an example of branching and merging that was tested during the course of this research. In this example, the sensor is designed to monitor the temperature of the clutch and send it to the switch, or the switch checks the temperature through the sensor and controls the electricity based on the temperature information. This example clearly shows the branching and merging scheme of components. The proposed software can handle such cases successfully.

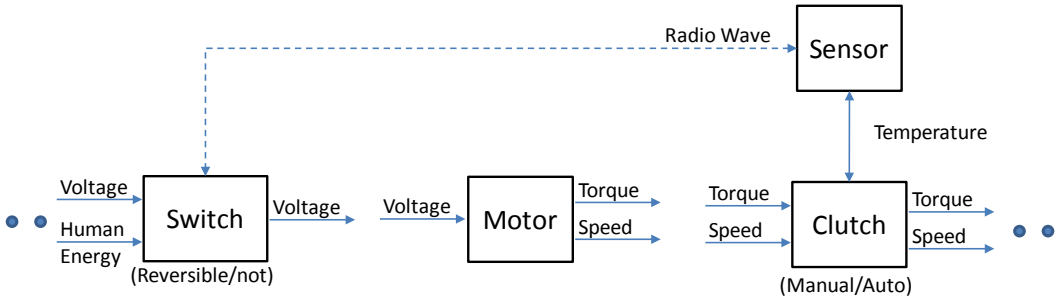

Figure 6-1. Example of branching and merging of Interfaces

The other main contributions of this work is defining the Interface and the development of the Machine Description Language (MDL), which enables component information to be collected from different suppliers in a standard way. Moreover, the MDL was developed in a machine-readable representation schema, thus enabling the proposed design software to work on

a more global component basis. The proposed system architecture references a hypothetical global database, but this can be effectively used in companies having large component libraries of their own. As observed in this application, even for a product composed of 7 components and a small library of components, it was a human-intensive process to manually find the cheapest feasible product – and manual searches many not always yield the cheapest product as was found in this work. So using the automated software to generate new products gives faster and more accurate results.

6.1.2 Limitations

The major limitation of the proposed modular product design software is its scalability. Since currently available data is small, the proposed software has not been tested in terms of its scalability. The solution (processing) time varies depending on not only the target objectives but also the size and "shape" of the feasible space traversed by the CPLEX IP solver based on the design requirements and available components. So the time taken to provide the solution becomes longer, when global manufacturers provide artifact information into the global repository.

As for other limitations, the proposed modular product design software can deal with quantitative measures, such as cost or weight, but it cannot handle qualitative measures, such as usability or convenience, unless they are quantified (e.g., via a rating scale). Additionally, the proposed modular product design software does not consider geometric constraints such as width, depth, and height of a module or component. In many cases, a product is required to be designed to fit a specific structure. Thus, a way to incorporate geometric constraints, starting with volume constraints, into the current model will be necessary for such applications.

The output from the tool is a single list of components, which is the optimum solution based on the objectives like cost, weight, etc., but for applications like concept generation, where

the objective is to present the user with multiple ranked feasible solutions, the automated concept generator is a better tool. So, a way of implementing some of the concepts like working directly from black box model into the automated concept generator needs to be investigated.

6.2 Future Work

Further work needs to be done to address some of the issues related to the modular product design software and to test some of its capabilities.

6.2.1 Output of the concept generator

The information on the wiki page should be integrated into the output of the concept generator. Also the method of presenting information like the specific design section on the component basis wiki pages needs to be further studied. The information on these wiki pages is not all encompassing; so, having different people working on it will improve the information available.

6.2.2 Interface

The performance type assigned to the flows in the definition of Interface was done by looking at the applications of the component terms and their flows. During the testing of the design tools built on the concept of Interface (like the modular product design software), as required new performance types can be assigned to the flows. The feature type was observed to be a field that is needed to fully specify an artifact, but without a clear definition of the type of

terms it should contain. This is one of the weak points of the Interface concept and needs to be examined further.

6.2.3 Modular product design software

Modular product design software that considers geometric constraints is needed to handle design requirements that require a product to fit within a certain geometric envelope. Since this formulation is based on Integer Programming, such geometric constraints can be easily added to the formulation. Also a more thorough analysis on the scalability issues should be made to find a robust heuristic method to solve modular product design problems in a reasonable amount of time. Although the ability to design a product family of screwdrivers was demonstrated using the proposed modular product design software, an approach to increase commonality across the product family is not incorporated into the model. Leveraging the proposed software to share modules and components within a family, the development of a methodology for product family design will be a valuable addition.

The capability of the tool to be used directly from the black box model (without the need to write the functional model) can also be validated as part of the survey. The user will be asked to represent his/her project both as a black box model and as a Functional Model. This black box model can be input into the modular product design software and the results can be analyzed.

Another direction to explore using this tool is a way to automatically build functional models from black box models similar to the grammatical construction of function structures [26] but by using the interface definition. This will greatly assist the concept generator by allowing it to accept black box models as input instead of the user creating a functional model (which is a major limitation of the concept generator as discussed).

Appendix A

Consolidated Component Naming Taxonomy

This Component Naming Taxonomy definition is based on the work by Kurtoglu, et al. [2]

Primary Component Classification	Secondary Component Classification	Component Term	Definition
Brancher	Separator	Abrasive	A device or material that uses texture on a surface to remove any portion of a firm (non-fluid) material.
		Blade	A device or material consisting of a broad flat or concave edge used to separate a firm (non-fluid) material.
		Centrifuge	A device that uses centrifugal force via a rapidly rotating container to separate fluid materials.
		Material Filter	A device or material consisting of a pattern of holes, slits, or pores used to separate constituents of a fluid mixture.
		Grater	
		Rake	A material filter that uses a series of parallel slits or tines to separate particles from the surrounding mixture.
		Sonicator	
		Stiff Brush	A device that uses bristles attached to a surface to remove any portion of a firm (non-fluid) material.
		Vibrator	A device that uses frequency oscillations to separate or dislodge a firm (non-fluid) material.
	Distributor	Distributor (Electric)	A device used to systematically allocate electrical energy along multiple paths.
		Nozzle	A device at the end of a pipe, hose, or tube used to distribute a continuous flow of fluid material.
		Soft Brush	A device that uses bristles to distribute a fluid material over a surface.
		Y-Pipe	
Channeler	Importer/Exporter	Electric Cord	A device used to bring electrical energy into a system from an external receptacle.
		Housing	A protective cover primarily used to bring flows into or out of a system that is also designed to contain or support components within it.
	Transferor	Belt	A device shaped as an endless loop of flexible material between two rotating shafts or pulleys used to transmit mechanical energy.
		Carousel	A device used to move material in a continuous circular path.
		Clutch	A device used to transmit rotational energy between two shafts that may be (dis)engaged smoothly.
		Conveyor	A device used to move material in a linear path.
		Electric Conductor	A device used to transmit electrical energy from one component to another.
		Electric Plate	An electric conductor in the form of a thin, flat sheet or strip.

		Electric Wire	An electric conductor in the form of a thin, flexible thread or rod.
		EM Transmitter	A device that transmits electromagnetic (EM) signals (such as infrared or RF) over a non-wired medium.
		Heat Exchanger	A device used to transmit heat from one medium to another.
		Rotational Coupler	A device used to connect coaxial shafts for power transmission from one to the other.
		Shaft	A device in the form of a cylindrical bar used to support rotating pieces or to transmit power or motion by rotation.
		Thermal Conductor	A device used to transmit thermal energy from one component to another.
		Tube	A device in the form of a hollow cylinder used to direct a fluid material (that is not under pressure) along a path.
	Guider	Bearing	A device in the form of a sphere or cylinder (or in an arrangement of spheres or cylinders) that is placed between moving parts to allow them to move easily relative to each other along a path.
		Hinge	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.
		Housing	A protective cover primarily used to bring flows into or out of a system that is also designed to contain or support components within it.
		Link	A device connecting two or more components that transmits motive power from one part to another along a specific path.
		Sled	A device either under or within a machine used to facilitate the sliding of components relative to each another along a path.
Connector	Coupler	Clamp	A device used to hold two or more components together that is readily (dis)engageable without the use of an external tool.
		Glue	A fastener in the form of an adhesive substance.
		Key	A fastener in the form of a piece of material that is inserted between other pieces, usually a pin-, bolt-, or wedge-like artifact fitting into a hole or space.
		Nut-Bolt	A fastener in the form of a threaded pin that screws into a usually square or hexagonal material through a threaded hole.
		Retaining Clip	A fastener in the form of a brace, band, or clasp.
		Rivet	A fastener in the form of a heavy pin having a head at one end with the other end hammered flat after being passed through holes in the joined pieces.
		Screw	A fastener in the form of a threaded pin, which does not require a nut to remain secure.
		Solder	A fastener in the form of a low-melting alloy used to join less fusible metals.
	Mixer	Agitator	A device used to maintain fluidity and plasticity, and to prevent segregation of liquids and solids in liquids, such as concrete and mortar.
		Carburetor	A device used to mix air with a fine spray of liquid fuel.
Magnitude Controller	Actuator	Door	A device in the form of a movable barrier, usually turning on hinges or sliding in a groove, and serving to close or open a passage into a space.
		Latch Release	A device that is designed to hold or free a mechanism as required.
		Electric Switch	A device for making or breaking the flow of electrical energy in an electrical circuit.
	Regulator	Grip Tape	
		Potentiometer	A device used to adjust the flow of electrical energy in an electric

			circuit.
		Thermostat	A device used to adjust temperature by starting or stopping the supply of heat.
		Transistor	A semiconductor device with three connections capable of regulating the flow of electrical energy in an electrical circuit.
		Valve	A device by which the flow of a fluid material may be adjusted by opening, shutting, or partially obstructing one or more ports or passageways.
		Varistor	A device used to adjust the flow of electrical energy in an electric circuit.
	Changer	Capacitor	A device used to alter a signal by storing an electrical charge.
		Choke	A device in the form of a restriction in a pipe that reduces the flow of a fluid material.
		Gear	A mechanical transformer in the form of a disc or plate that transmits mechanical energy to another device by means of teeth.
		Inclined Plane	A device in the form of a surface sloped at an angle to a reference surface, which provides a mechanical advantage for raising loads.
		Inductor	A device used to alter a signal by storing energy as a magnetic field.
		Lens	A device in the form of a translucent substance used to alter the path of optical energy transmitted through it.
		Lever	A device fixed at a fulcrum and acted on at two other points by two forces, each tending to cause it to rotate in opposite directions round the fulcrum.
			A mechanical transformer in the form of a disc or plate that transmits mechanical energy to another device by means of teeth.
		Mold	A hollow device used to give shape to a molten or hot fluid when it cools and hardens.
		Needle	A device in the form of a slender, usually pointed, rod used to amplify a mechanical rotation on a dial or other measuring instrument.
		Pulley	A mechanical transformer in the form of a wheel or drum fixed on a shaft and turned by a belt, chain, or strap.
		Punch	A device used to make holes, impress a design, or stamp a die into a firm material.
		Resistor	A device that alters the flow of electrical energy by resisting the passage of electrical current.
		Sprocket	A mechanical transformer in the form of a toothed wheel that engages a power chain.
		Transformer	A device that alters the flow of mechanical or electrical energy during the process of transmitting it.
	Stoppers	Acoustic tile	A device used to prevent the passage of sound, or vibration.
		Cap	A device in the form of a firm material secured to and used to prevent the flow of material into a hole or aperture.
		Check Valve	A device that allows a fluid to flow in only one direction.
		Cover	A device that overspreads an object, which is used to hide, defend, or shelter a material.
		Cushion	A device in the form of a soft pad or bumper used to prevent the transmission of mechanical energy from jarring, friction, or pressure.
		Diode	A semiconductor device which allows current to flow in only one direction.
		Electric Insulator	A device used to prevent the passage of electrical energy.
		Fuse	A device that breaks the flow of electrical energy in an electrical circuit in response to an excessive current.

		Muffler	
		Seal	A device used to prevent the flow of a fluid material, especially at a place where two surfaces meet.
		Stop	A device in the form of a rigid structure that is automatically activated by a predetermined displacement to limit the operation of a system.
		Thermal Insulator	A device used to prevent the passage of thermal energy.
Converter	Material Converter	Catalytic Converter	A device used to chemically transform a harmful gas material into one or more inert forms.
		Condenser	A device used to transform a gas material into a liquid material.
		Evaporator	A device used to transform a liquid material into a gas material.
	Energy Converter	Airfoil	A device with curved surfaces used to transform pneumatic energy into translational energy.
		Armature	A device used to transform magnetic energy into rotational energy.
		Bulb	A device used to transform electrical energy into the spectrum of electromagnetic energy visible to humans.
		Burner	A device used to transform chemical energy into thermal energy.
		Cam	A device in the form of an eccentric curved wheel or disc used to transform rotational energy into reciprocating translational energy.
		Crank	A device used to transform reciprocating translational energy into rotational energy.
		Electric Motor	A device used to transform electrical energy into mechanical energy.
		Electromagnet	A device used to transform electrical energy into magnetic energy.
		Fan	A device in the form of a rotating shaft with two or more broad, angled blades attached used to transform rotational energy into pneumatic energy.
		Generator	A device used to transform mechanical energy into electrical energy.
		Heating Element	A device used to transform electrical energy into thermal energy.
		Hydraulic Piston	A device in the form of a cylinder tightly fitted inside a tube used to transform hydraulic energy into translational energy.
		Hydraulic Pump	A device used to transform mechanical energy into hydraulic energy by altering the pressures within a system.
		IC Motor	A device used to transform chemical energy in the form of liquid fuel into mechanical energy.
		Pneumatic Piston	A device in the form of a cylinder tightly fitted inside a tube used to transform pneumatic energy into translational energy.
		Pneumatic Pump	A device used to transform mechanical energy into pneumatic energy by altering the pressures within a system.
		Screw Propeller	A device in the form of a rotating shaft with two or more broad, angled blades attached used to transform rotational energy into hydraulic energy.
		Speaker	A device used to transform an electrical signal into acoustic energy.
		Wheel	A device in the form of a disc or circle used to transform translational energy applied at the hub into rotational energy.
	Signal Converter	Knob	A device used to transform human energy into a control signal.
Provisioner	Material Supplier	Bladder	A device in the form of a hollow, expandable sac or membrane with a narrow opening used to accumulate and dispense a material.
		Container	A device in the form of a closed canister used to accumulate and dispense a material.

		Pressure Vessel	A device in the form of a sealed tank used to accumulate and dispense a pressurized fluid material.
	Energy Supplier	Battery	A device used to accumulate and dispense electrical energy by means of a chemical reaction.
		Flywheel	A device used to accumulate and dispense rotational energy via angular momentum.
		Spring	A device used to accumulate and dispense mechanical energy via the elastic properties of the device's material properties.
Signaler	Sensor	Acoustic Sensor	
		Ammeter	A device used to determine the current through an electric circuit.
		Displacement Gauge	A device used to determine translational or rotational distance in a system.
		EM Sensor	A device used to detect an electromagnetic signal.
		Level Gauge	A device in the form of an external plate or face on which the amount of a fluid material is determined.
		Pressure Gauge	A device used to determine the pressure from hydraulic or pneumatic energy in a system.
		Speed Gauge	A device used to determine velocity in a system.
		Timer	
		Voltmeter	A device used to determine the voltage across a portion of an electric circuit.
	Indicator	Analog Display	A visual indicator in the form of a continuously variable dial or gauge.
		Bell	An auditory indicator in the form of a hollow object that is struck to produce vibration.
		Buzzer	An auditory indicator in the form of an electronic device that emits a buzzing noise.
		Digital Display	A visual indicator in the form of a discrete readout or gauge.
		Recording	An auditory indicator in the form of stored acoustic information that is replayed.
Supporter	Stabilizer	Insert	A device in the form of a material around which another material sets, solidifies, or is formed and used to strengthen or prevent a material from overturning.
		Support	A device that holds up or sustains the weight of a body.
	Securer	Bracket	A device in the form of a piece or combination of pieces, usually triangular in general shape, projecting from, or fastened to, a wall, or other surface, to secure heavy bodies or angles.
	Positioner	Handle	A device used to place a human hand in an appropriate configuration for grasping or interacting.
		Washer	A device in the form of a disk or ring used to provide spacing between components located on a axle or shaft.

Appendix B

Artifact Data for Product Family of Screwdrivers

Motor						
ID	Company Name	Price	Weight	Input Volt	Torque	Speed
1	Black and Decker	2.20	0.50	1	3	4
31	Xmoter	3.82	0.75	1	5	6
41	Hobbyzone	4.02	0.30	1	3	4
61	Hobbyzone	7.18	0.20	1	3	4
71	Drevolution	7.40	1.12	1	3	4
141	M-1	1.50	0.11	8	10	11
151	M-2	2.24	1.15	9	3	4

Switch							
ID	Company Name	Price	Weight	Input Volt	Input Joules	Output Volt	Feature Reversibility
2	Black and Decker	0.14	0.10	1	2	1	yes
42	Eswitch	0.84	0.04	1	2	1	yes
72	Drevolution	0.98	0.20	1	2	1	yes
112	S-1	0.11	0.02	1	2	1	no
142	S-3	0.61	0.22	8	2	8	yes
152	S-2	0.60	0.20	8	2	8	no

Battery Pack						
ID	Company Name	Price	Weight	Input Volt	Volt	Feature High/Low
3	Black and Decker	0.72	0.90	1	1	yes
23	SamJung	0.32	0.92	8	8	yes
53	SamJung	0.88	0.95	9	9	yes
73	Drevolution	0.88	1.01	8	8	yes
113	B-1	0.62	0.50	1	1	no
133	B-3	0.24	0.95	8	8	no
143	B-2	0.87	0.98	8	8	yes

Voltage Convertor					
ID	Company Name	Price	Weight	Input Vdt	Output Vdt
4	Black and Decker	1.02	0.05	1	1
24	SamJung	1.22	0.84	8	1
54	Drevolution	1.42	0.41	9	1
74	Drevolution	1.52	0.92	8	1
134	V-1	0.78	0.91	8	1
154	V-2	1.43	0.88	9	1

Gear Box							
ID	Company Name	Price	Weight	Input Torque	Speed	Output Torque	Speed
5	Black and Decker	1.02	0.40	3	4	5	6
65	JoongAng	1.94	0.13	3	4	5	6
75	Drevolution	2.32	0.83	3	4	5	6
145	G-3	2.01	0.47	10	11	5	6
155	G-1	0.98	0.10	3	4	10	11
165	G-2	1.13	0.61	5	6	3	4

Clutch								
ID	Company Name	Price	Weight	Input Torque	Speed	Output Torque	Speed	Feature Mode Select/Not
6	Black and Decker	0.70	0.20	5	6	5	6	yes
46	Boom	1.92	0.02	5	6	5	6	yes
76	Drevolution	2.43	0.52	5	6	5	6	yes
126	C-1	0.50	0.01	5	6	5	6	no
156	C-2	2.80	0.21	10	11	5	6	yes
166	C-3	2.81	0.23	10	11	10	11	yes

Sleeve								
ID	Company Name	Price	Weight	Input Torque	Speed	Output Torque	Speed	Performance Interface Type
7	Black and Decker	0.31	0.3	5	6	5	6	7
47	FSleeve	0.97	0.02	5	6	5	6	7
77	Drevolution	1.18	0.46	5	6	5	6	7
157	SL-1	0.28	0.01	5	6	5	6	12
167	SL-2	0.50	0.15	10	11	10	11	7

Appendix C

Integer Programming Formulation for Modular Product Design [5]

Domain definition

P is the set of parts published in the cyber-infrastructure.

I is the set of all Interfaces of parts in P.

F is the set of all features of parts in P.

$I_{In} \subseteq I$ is the set of Interfaces that are used as input to any part.

$I_{Out} \subseteq I$ is the set of Interfaces that are used as output in any part.

$F_{Out} \subseteq F$ is the set of features that are obtained from any part.

$I_{Initial} \subseteq I$ is the set of Interfaces that are initially given.

$I_{Goal} \subseteq I$ is the set of Interfaces that are initially given.

$F_{Goal} \subseteq F$ is the set of goal features.

$P_i^{input-consumed} \subseteq P, \forall i \in I$ is the set of parts that have Interface i as input and the Interface is consumed(used).

$P_i^{input-unconsumed} \subseteq P, \forall i \in I$ is the set of parts that have Interface i as input and the Interface is not consumed(used).

$P_i^{output} \subseteq P, \forall i \in I$ is the set of parts that have Interface i as output.

$P_f^{output} \subseteq P, \forall f \in F$ is the set of parts that have feature f as output.

SF_s is the set of features obtained until Stage s.

Stage (s): $1 \le s \le S$, where S is the maximal number of stages for modular product design.

P_s is the set of parts simultaneously used in product design at Stage s.

Variable Definition

For all $p \in P$, $s \in 1,...,S$, $y_{p,s}$'s are *part usage variables*.

$$y_{p,s} = \begin{cases} 1 \text{ if part } p \text{ is used in stage } s, \\ 0 \text{ otherwise.} \end{cases}$$

The following variables are *Interface or feature usage variables*.

$$x_{i,s}^{available-unused} = \begin{cases} 1 \text{ if interface } i \text{ is availabe but not used in stage } s, \\ 0 \text{ otherwise.} \end{cases}$$

$$x_{i,s}^{input-consumed} = \begin{cases} 1 \text{ if part } p \text{ is used in stage } s \text{ such that } p \in P_i^{input-consumed}, \\ 0 \text{ otherwise.} \end{cases}$$

$$x_{i,s}^{input-unconsumed} = \begin{cases} 1 \text{ if part } p \text{ is used in stage } s \text{ such that } p \in P_i^{input-unconsumed}, \\ 0 \text{ otherwise.} \end{cases}$$

$$x_{i,s}^{output} = \begin{cases} 1 \text{ if part } p \text{ is used in stage } s \text{ such that } p \notin P_i^{input} \wedge p \in P_i^{output}, \\ 0 \text{ otherwise.} \end{cases}$$

$$x_{f,s}^{feature} = \begin{cases} 1 \text{ if part } p \text{ is used in stage } s \text{ such that } f \in F, \\ 0 \text{ otherwise.} \end{cases}$$

Formulation

(1) Objective Function

Any numerical expression can be defined for this objective function based on the characteristics of targeting market segments, weight, or price can be the objective. Furthermore, multiple objectives can be defined. In such case, goal programming or other multi-criteria optimization methodologies can be utilized.

$$Minimize \sum_{w \in W} \sum_{s \in S} c_p \cdot y_{p,s}, \text{ where } c_p \text{ is the cost for the use of part } p.$$

(2) Initial constraints

The initial input Interfaces are expressed by setting 1 to output Interface usage variables at Stage 0.

$x_{i,0}^{output} = 1,\ x_{i,0}^{input-unconsumed} = x_{i,0}^{available\text{-}unused} = x_{i,0}^{input\text{-}consumed} = 0 \ \ \forall i \in I_{Initial}$: Given Interfaces at initial stage

$x_{i,0}^{input-unconsumed}, x_{i,0}^{input-consumed}, x_{i,0}^{output}, x_{i,0}^{available\text{-}unused} = 0, \forall i \notin I_{Initial}$: Other Interfaces at initial stage

$x_{f,0}^{output} = x_{f,0}^{available-unused} = 0, \forall f \in F$: All features at initial stage

(3) Goal constraints

The goal of modular product design is represented in goal constraints. If the goal features are used as output variables at the last stage, the goal is achieved.

$x_{f,S}^{output} + x_{f,S}^{available-unused} \geq 1 \ \ \forall f \in F_{Goal}$: Goal features at the final stage

$x_{i,S}^{input-unconsumed} + x_{i,S}^{input-consumed} + x_{i,S}^{output} + x_{i,S}^{available-unused} \geq 1 \ \ \forall i \in I_{Go\ a}$: Goal Interfaces at the final stage.

(4) Compatibility constraints

The compatibility constraints play a role of making the functional requirements satisfied.

$$\sum_{p \in P_i^{output}} y_{p,s} \geq x_{i,s}^{output} \ \ \forall i \in I, s \in 1,...,S$$

$$y_{p,s} \leq x_{i,s}^{output} \ \ \forall p \in P_i^{output}, \forall i \in I, s \in 1,...,S$$

$$\sum_{p \in P_i^{input-unconsumed}} y_{p,s} \geq x_{i,s}^{input-unconsumed} \ \ \forall i \in I, s \in 1,...,S$$

$$y_{p,s} \leq x_{i,s}^{input-unconsumed} \ \ \forall p \in P_i^{input-unconsumed}, \forall i \in I, s \in 1,...,S$$

$$\sum_{p \in P_f^{output}} y_{p,s} \geq x_{f,s}^{output} \ \forall f \in F, s \in 1,...,S$$

$$y_{p,s} \leq x_{f,s}^{output} \ \forall p \in P_f^{output}, \forall f \in F, s \in 1,...,S$$

$$\sum_{p \in P_i^{input-consumed}} y_{p,s} = x_{i,s}^{input-consumed} \ \forall i \in I, s \in 1,...,S$$

(5) Non-concurrency constraints

Once Interface *i* is decided to be used as an input-unconsumed, input-consumed or output variable, which means $x_{i,s}^{input-unconsumed}$, $x_{i,s}^{input-consumed}$ or $x_{i,s}^{output}$ is set to 1, respectively, $x_{i,s}^{available-unused}$ should not be used. In other words, it should not be set to 1.

$$x_{i,s}^{output} + x_{i,s}^{available\text{-}unused} + x_{i,s}^{input\text{-}consumed} \leq 1 \ \forall i \in I, s \in 1,...,S$$

$$x_{i,s}^{input-unconsumed} + x_{i,s}^{available\text{-}unused} + x_{i,s}^{input\text{-}consumed} \leq 1 \ \forall i \in I, s \in 1,...,S$$

(6) Sequence constraints

Only when Interface *i* is used as an output in previous stages, it can be used as input or available-unused variables in the next stage. Such sequential requirements are represented in the sequence constraints.

$$x_{i,s}^{input-unconsumed} + x_{i,s}^{available\text{-}unused} + x_{i,s}^{input\text{-}consumed} \leq x_{i,s-1}^{input-unconsumed} + x_{i,s-1}^{available\text{-}unused} + x_{i,s-1}^{output} \ \forall i \in I, s \in 1,...,S$$

(7) Binary variables

Here are all the variables defined in the formulation.

$$x_{i,s}^{input-unconsumed}, x_{i,s}^{input-consumed}, x_{i,s}^{output}, x_{i,s}^{available\text{-}unused} \in \{0,1\} \ \forall i \in I, s \in 1,...,S$$

$$x_{f,s}^{output}, x_{f,s}^{available\text{-}unused} \in \{0,1\} \ \forall f \in F, s \in 1,...,S$$

$$y_{p,s} \in \{0,1\} \ \forall p \in P, s \in 1,...,S$$

Bibliography

[1] Bryant, C. R., McAdams, D. A., Stone, R. B., Kurtoglu, T. and Campbell, M. I., 2005, "A Computational Technique for Concept Generation," *ASME International Design Engineering Technical Conference – Computers and Information in Engineering Conference*, September 24-28, Long Beach, CA, DETC2005-85323.

[2] Kurtoglu, T., Campbell, M. I., Arnold, C. B., Stone, R. B., and McAdams, D. A., 2009, "A Component Taxonomy as a Framework for Computational Design Synthesis," *Journal of Computing and Information Science in Engineering*, 9.

[3] Google Sketchup, April 9, 2010, http://sketchup.google.com/

[4] Component Basis, March 21, 2010, http://www.edesignonline.org/wiki/index.php?title=Component_Basis

[5] Yoo, J., Kumara, S., and Simpson, T. W, 2009, "Modular Product Design for Global Manufacturing Using Cyberinfrastructure," *ASME Design Engineering Technical Conferences – Computers and Information in Engineering Conference*, August 30 - September 2, San Diego, CA, DETC2009-86899.

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

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

[8] Ahmed, S., and Wallace, K., 2003, "Evaluating a Functional Basis," *Proc. ASME Design Engineering Technical Conferences - Design Theory and Methodology*, Chicago, IL.

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

[10] Wang, E., Kim, Y. S., and Kim, S., 2005, "An Object Ontology Using Form-Function Reasoning to Support Robot Context Understanding," *Computer-Aided Design and Applications*, 2(6), pp. 815-824.

[11] Tinsley, A., Midha, P. A., Nagel, R. L., McAdams, D. A., Stone, R. B., and Shu, L. H. , 2007, "Exploring the Use of Functional Models as a Foundation for Biomimetic Conceptual Design," *International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, September 4-7, Las Vegas, Nevada, USA, DETC2007-35604.

[12] Chandrasekaran, B., Stone, R. B., and McAdams, D. A. , 2004, "Developing Design Templates for Product Platform Focused Design," *Journal of Engineering Design*, 15(3), pp. 209-228.

[13] Bryant, C. R., Bohm, M. R., Stone R. B. and McAdams D. A., 2007, "An Interactive Morphological Matrix Computational Design Tool: A Hybrid of Two Methods," *ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, September 4-7, Las Vegas, Nevada, DETC2007/DTM-35583.

[14] Pahl, G., and Beitz, W., 1996, *Engineering Design - A Systematic Approach*, Springer, London.

[15] 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.

[16] Bohm, M. R., Stone, R. B., Simpson, T. W. and Steva, E. D., 2008, "Introduction of a Data Schema to Support a Design Repository," *Computer-Aided Design*, 40(7), pp. 801-811.
[17] Bohm, M. R., and Stone, R. B., 2004, "Product Design Support: Exploring a Design Repository System," *ASME International Mechanical Engineering Congress & Exposition*, Anaheim, CA, IMECE2004-61746.
[18] Szykman, S., and Sriram, R., 2006, "Design and Implementation of the Web-Enabled Nist Design Repository," *ACM Transactions on Internet Technology*, 6(1), pp. 85-116.
[19] Design Engineering Lab Repository, March 22, 2010, http://function2.mime.oregonstate.edu:8080/view/browse.jsp
[20] Rao, T. P. L., 2010, "Enhancing the Visulaization Tools of a Concept Generator," MS Thesis, Pennsylvania State University, University Park, PA.
[21] Godthi, V., Yoo, J., Bryant, C. R., Simpson, T. W., and Kumara, S. R. T., 2010, "Product Design Using Interface Based Module Description."
[22] XML, February 4, 2010, http://www.w3.org/XML/
[23] Szykman, S., Senfaute, J. and Sriram, R., 1999, "The Use of Xml for Describing Functions and Taxonomies in Computer-Based Design," *ASME Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, Las Vegas, NV, DETC1999/CIE-9025.
[24] CPLEX, August 20, 2009, http://www.ilog.com/products/cplex/
[25] Marion, T. J., and Simpson, T. W., 2006, "Platform Leveraging Strategies and Market Segmentation," In T. W. Simpson, Siddique, Z., and Jiao, R. J., *Product Platform and Product Family Design*, Springer, New York, USA, Chap. 5, pp. 73-90.
[26] Sridharan, P., and Campbell, M. I., 2005, "A Study on the Grammatical Construction of Function Structures," *Artificial Intelligence in Engineering Design, Analysis, and Manufacture*, 19, pp. 139-160.