A computational approach to innovative conceptual design

1 author:

Tolga Kurtoglu
Palo Alto Research Center
67 PUBLICATIONS **1,204** CITATIONS

SEE PROFILE

The Dissertation Committee for Tolga Kurtoglu Certifies that this is the approved version of the following dissertation:

A COMPUTATIONAL APPROACH TO INNOVATIVE CONCEPTUAL DESIGN

Committee:

Matthew I. Campbell, Supervisor

Benito Fernandez-Rodriguez

Bruce Porter

Robert B. Stone

Kristin L. Wood

A COMPUTATIONAL APPROACH TO INNOVATIVE CONCEPTUAL DESIGN

by

Tolga Kurtoglu, BS, MS

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August, 2007

Dedication

I dedicate this work to my wife Bige.

A COMPUTATIONAL APPROACH TO INNOVATIVE CONCEPTUAL DESIGN

Publication No._____

Tolga Kurtoglu, PhD

The University of Texas at Austin, 2007

Supervisor: Matthew I. Campbell

Conceptual design is a vital part of the design process during which designers first envision new ideas and then synthesize them into physical configurations that meet certain design specifications. In this research, a computational approach is developed to assist the designers perform this non-trivial task of navigating the design space for creating conceptual design configurations. The methodology is based on combining empirical reverse engineering techniques with a graph-grammar approach. Accordingly, design knowledge is systematically extracted from past designs, formulated as procedural grammar rules, and employed in building new design concepts. The implemented system provides a theoretical framework for automatically searching conceptual design spaces and produces novel alternative configurations to real design problems. The application of the approach to the design of various electromechanical devices shows the method's range of capabilities, and how it serves as a comparison to human conceptual design generation and as a tool to complement the skills of a designer.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Product design is an iterative decision-making process for transforming a design opportunity into an embodied product that satisfies a set of requirements. It begins with identification of a need, proceeds through a series of tasks to seek an optimal solution to the problem, and ends with a detailed description of the product to be manufactured for the customer. The product development process, at its highest level, is characterized with three phases: problem identification, conceptual design, and embodiment or detailed design. In the first phase, information about the product is collected and the design problem is defined. The second phase encompasses all activities to generate physical solutions to meet design specifications. The third phase is where decisions are made to finalize dimensions, shape, material properties, and layout of the design. At the end of this phase, a product is fully specified and ready to be manufactured.

Among these phases, conceptual design is the main focus of this research. Specifically, the purpose of the research is to examine the conceptual design stage and develop a systematic method and resulting computer-based tools for improving the designer's ability to create or invent design solutions.

1.1 INTRODUCTION TO CONCEPTUAL DESIGN

This section describes the conceptual phase of product development and presents structured approaches to conceptual design and conventional practices used for concept generation.

1.1.1 Overview of the Conceptual Design Process

Conceptual design plays the central role in ensuring the overall design quality and the level of innovation. It is at this phase, where the architecture of the final design is

established, the technologies are chosen to fulfill the customer needs, and when most of the cost of a product is committed. Because of these characteristics, conceptual design is often considered as the most important phase of the product development cycle.

The goal of conceptual design is to generate ideas that address a set of requirements. The conceptual design process begins with the specification of the product to be designed and involves the continual cycle of concept generation and concept evaluation as shown in Figure 1.1. In the first step, all possible concepts that could address the design specifications are generated. This is followed by an evaluation process carried out to select the best candidate for further design refinement and embodiment. Often times, these two steps overlap and the boundaries between them are unclear. Yet, they are useful to categorize the tasks needed for conceptual design and are also utilized in the development of the proposed method and the organization of this dissertation.

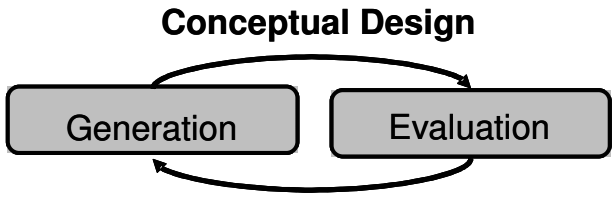


Figure 1.1 The two processes that characterize conceptual design

1.1.2 Review of Structured Design Methods

The generation phase of conceptual design is difficult to translate into a concise methodology. Concept generation, fundamentally, is considered an informal, highly creative artistic activity, not a formal, scientific endeavor. Historically, the only resource available to a designer during conceptual design was personal repertoire of design knowledge. Therefore, the ability to apply creativity and to invent has been heavily dependent on a designer's expertise.

Recently, however, increasing attention is being directed to support conceptual level design activities, and to develop methods that quantify and formalize the conceptual phase of design (Pahl and Beitz, 1988; Suh, 1990; Ulrich and Eppinger, 1995; Otto and Wood, 2001; Ullman 1995; Hubka and Eder 1984). Regardless of the methodology variation, all of these approaches begin by formulating the overall product function and breaking it into smaller, more elemental sub-functions. Solutions to these sub-functions are sought and the form of the device then follows from the assembly of all sub-function solutions. This process is illustrated in Figures 1.2 - 1.5 and is often referred to as the mapping of function to form.

In this systematic view of conceptual design, a designer formulates the overall function by analyzing the specifications of the product to be designed. An example of this formulation is given in Figure 1.2 for the design of an electric toothbrush product. This high level product function is then decomposed recursively into lower level functions - a process that produces a function structure (Pahl and Beitz, 1988) which is a representation that defines function as transformation between input and output of energy, material, and information (elaborated in Chapter 4). The function structure for the electric toothbrush is presented in Figure 1.3. This functional decomposition is then used to generate form solutions to each of the product sub-functions. Here, the designer seeks solutions, i.e. a component or a set of components that perform a particular function as shown in Figure 1.4. Finally, solutions to the sub-functions are synthesized together to arrive to the final form of a product completing the transformation of function to form. The final conceptual configuration of the electric toothbrush product is shown in Figure 1.5.

STEP1: transform customer needs into an overall product function

Figure 1.2 The first step of systematic concept generation process

STEP 2: decompose function into elemental sub-functions using a repeatable functional representation

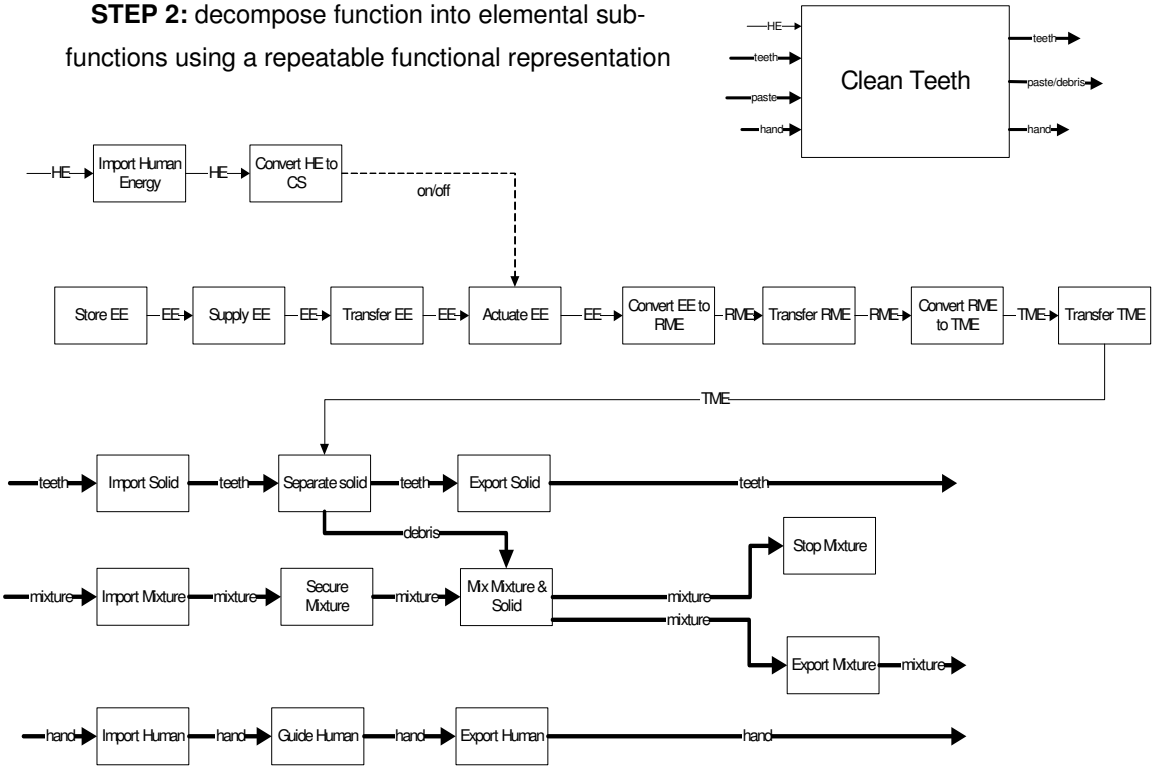


Figure 1.3 The second step of systematic concept generation process

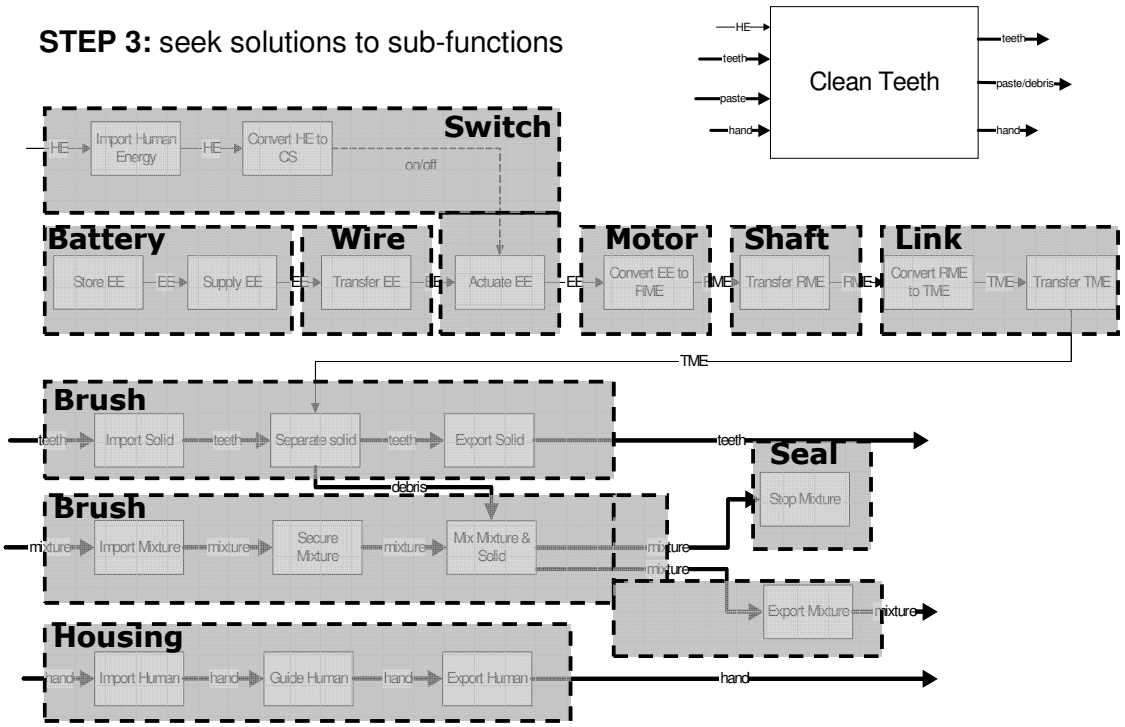


Figure 1.4 The third step of systematic concept generation process

STEP 4: synthesize elemental solutions to arrive at a system level conceptual configuration

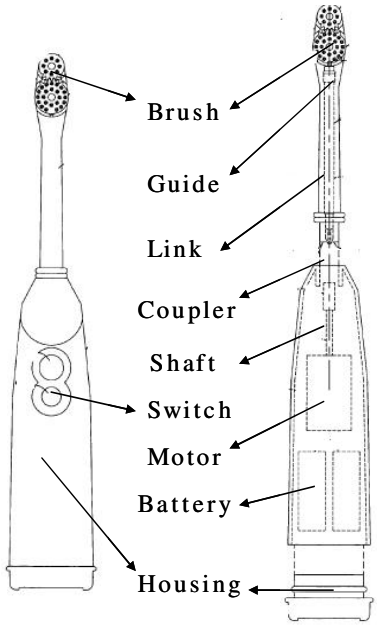


Figure 1.5 The fourth step of systematic concept generation process

Using this approach, a broad number of concepts can be generated by selecting and composing different design solutions to elemental sub-functions. A number of non-computational methods are available to help designers create ideas during this process. These are reviewed in the next section.

1.1.3 Review of Human-Based Concept Generation Methods

Concept generation research has traditionally focused on developing methods that improve the quality and variety of concepts generated. These methods are often kept simple and efficient such that designers are not burdened by the details or limitations of the method. The most common concept generation method is known as brainstorming (Osborn, 1957). The term brainstorming is frequently applied to any idea generation technique. Brainstorming as a specific method requires a group of individuals to follow the basic rules of (1) avoiding criticism, (2) welcoming "wild ideas", (3) building on one another's ideas, and (4) preferring more ideas than dwelling on specific ones.

A more structured concept generation method can be found in the techniques known as C-Sketch (Shah, 1998) and 6-3-5 (Rochbach, 1969). The latter of these sketch-based methods requires six participants to independently create three ideas at a time in a series of five rounds. The added constraints of the method ensures that individuals participate equally which may be difficult to enforce in traditional brainstorming.

In addition to these group methods to concept generation, there are also some well accepted approaches that do not require a set of interacting designers. Designing by analogy (McAdams and Wood, 2000) is a well accepted approach to arrive at novel design solutions. It can be accomplished by first generalizing the design problem to a set of functions (or a graph of functions as in the function structure representation). Then one can look for or conceive analogous products or components that perform the same set of functions (Linsey et al., 2005). Function-means trees and Morphological Analysis

(Zwicky, 1969) are similar methods in which solutions to individual functional requirements are first sought and then synthesized together.

Apart from these approaches, one widely used method is the Theory of Inventive Problem Solving (Altschuller, 1984). This method provides a tabulated representation of a large number of solution principles that have been extracted from existing patents.

Another approach is "catalog design" where concepts are generated purely through browsing a catalog of physical elements (components, assemblies, etc.). The results are evidently limited by the breadth of the catalog; however, the benefit lies in the presentation of design knowledge that falls outside the designer's expertise memory (McAdams and Wood, 2001).

1.2 VISION AND GOAL OF THE RESEARCH

As summarized in Section 1.1.2, many systematic approaches have emerged in recent years to help guide designers during the conceptual stage of design (Pahl and Beitz, 1988; Suh, 1990; Ulrich and Eppinger, 1995; Otto and Wood, 2001; Ullman 1995; Hubka and Eder 1984). Yet, the conceptual design process has seen few attempts at automation. The concept of "automating design" has often been leveraged in later stages of the design process where a to-be-designed artifact accrues numerous parameters but lack specific dimensions. Automated methods such as optimization provide a useful framework for managing and determining details of the final designed artifact. These methods make the design process less tedious and time-consuming and are used in a wide variety of industries to support or optimize current design efforts. However, one of the pervasive bottlenecks in design is the lack of continuity between computational design tools and conceptual design methods.

The goal of this research is to bridge this gap. In order to achieve this goal, the conceptual design stage is examined with the aim of understanding and formalizing the

early phases of the design process, and a new conceptual design theory is proposed by developing an automation tool that integrates characteristics of human conceptual designing with a computational synthesis technique.

The motivation for developing this theory are as follows: (1) to fully automate the systematic design process described in Section 1.1.2, and thereby to create a computational tool that can automatically generate solutions to open-ended design problems, (2) to broaden the applicability of computers in the early phases of the design process for more efficient, creative, and innovative conceptual design, (3) to create a basis for future CAD technology that can computationally represent the highly abstract, function-oriented design knowledge relevant to conceptual design, and to provide a framework that enables the integration of this knowledge for design generation and evaluation at the early stages, and (4) to computationally model the decision-making employed during conceptual design that is linked to human creativity and invention.

1.3 HYPOTHESIS

As stated above, the primary goal of this research is to automate the function-based synthesis paradigm introduced by the systematic design process (Pahl and Beitz, 1988). Based on this process, the creative synthesis step of conceptual design can be summarized by two important questions:

1. Which solutions should be used in a concept for a given sub-function?
2. How should the generated solutions for individual sub-functions be synthesized together into a final configuration?

In answering these questions, there are many instances where the designer makes conscious and unconscious decisions to select and refine concept variants in order to map "function" to "form". As described earlier, the success of this transformation and its resulting artifact are subject to the experience and talents of the designer involved.

Fortunately, experience in the form of design knowledge is implicitly available in actual examples of designers' work. Heuristics-based decisions made by designers in transitioning from functional requirements to conceptual design configurations can be extracted by studying existing products. In this research, it is conjectured that through systematic dissection of consumer products and a thorough reverse engineering process, a methodology can be developed that extracts design knowledge employed in the creation of designs. Accordingly, this research leverages a database of past products from which a *design grammar* is developed to capture the knowledge of the original designer in transforming designs from function to form. The *grammar rules* created from the empirical analysis of existing products are then used in a computational search process. The resulting computational design tool answers the two aforementioned questions that define the creative synthesis step of conceptual design, and works with a designer in navigating the design space to create design configurations from detailed specifications of product function. Given the overwhelming size of the feasible solution space, such a computational tool:

1. Increases the efficiency of the design process and the creation of new solutions,
2. Facilitates design reuse during concept generation,
3. Enables the exploration of larger design solution spaces, and thereby removes psychological bias that may limit designers to previous solutions or to specific engineering domains.

1.4 RESEARCH OBJECTIVES

It is the objective of this research to develop computational design tools to compute design alternatives. The proposed approach describes the comprehensive space of concept variants and searches it for feasible candidates. With these design tools, a

design team is able to rapidly generate multiple and feasible design configurations. The configuration description includes the choice of components (component selection - Step 3 illustrated in Figure 1.4) and how they are connected (component configuration - Step 4 illustrated in Figure 1.5). In addition, the design tools provide the necessary means for a design team to evaluate overall concepts based on specific design objectives. The implemented system serves as a comparison to human conceptual design generation and as a tool to complement the skills of a designer or a design team.

1.5 THESIS STATEMENT

A conceptual design theory based on an empirical study of existing artifacts and a graph-grammar-based computational search creates concept variants from product function specifications.

1.6 ORGANIZATION

This dissertation presents a full description of the developed conceptual design theory and the resulting computational design tools. The document can be thought of as three major sections. The early chapters (Chapters 1-4) present the fundamental elements leading to the development of the theory, the following three chapters (Chapters 5-7) focus on the main deliverables of this research that present the computational design method, and the design tools for concept generation and evaluation, and finally the later chapters (Chapter 8-10) are devoted to the validation of the proposed method and provide example test problems and experimental studies. More specifically;

Chapter 2 provides an overview of the research approach and presents the procedure of developing the computational design framework.

Chapter 3 discusses the tools and methods utilized during product teardowns and subsequent knowledge acquisition.

Chapter 4 introduces the method for representing the designs. It provides an overview of the representation of function and form in design, and the specifics of the graph-based representations used by the computational method in particular.

Chapter 5 describes the development of a graph grammar for conceptual design. It includes a discussion of fundamentals of graph grammars, grammar derivation, and the description of the process used in developing the grammar rules for conceptual design.

Chapter 6 details the generation aspects of the developed computational design method. It describes the automated concept generation process, and a summary of computer-based implementation of the grammar and the graphical user interface that is used.

Chapter 7 introduces the evaluation strategy.

Chapter 8 presents the results of applying the computational method on two electromechanical test problems.

Chapter 9 discusses the experiments that are developed to assess the effects of using the computational design tool as a conceptual design aid.

Chapter 10 provides a summary of the research and a discussion of the contributions and potential venues for future work.

Chapter 2: Automated Synthesis of Design Concepts from Empirical Analysis of Function to Form Mapping

In this chapter, an overview of developing the grammar-based computational design framework is presented. This development involves three major processes: design knowledge extraction via empirical product analysis, representation of design knowledge, and computational design synthesis.

In the first process, product information is accumulated by means of product dissections, and empirical product analysis. This product information is collected for a particular category of products. The central part of the process is the function-based and configuration-based product information models, which will be elaborated in Chapters 3 and 4.

The next process is the representation of the design knowledge. Here, the information collected in the first process is represented computationally. There are three distinct knowledge representation schemes used as part of this research. One of them is adopted from the design literature (functional representation), and two of them are specifically developed for the purposes of this research (configuration representation, and function-to-configuration mapping representation).

Finally, the computational design synthesis process deals with creating conceptual design solutions. Initially, a design problem is put forth as functional design specification consistent with the language used for the function-based product models. By using and combining existing product information, this design specification is converted to a configuration-based product model which is presented at the end as final conceptual solution.

2.1 AUTOMATING THE FUNCTION-BASED SYNTHESIS PROCESS

As described in Chapter 1, this research aims to automate the systematic design process (Pahl and Beitz, 1988). Accordingly, it extends the previous function-based design research by developing a computational design method that transforms a high-level, functional description of a non-existent product into a set of concept variants. Furthermore, it presents an automated process that can estimate the worth of such concept variants based on different evaluation criteria.

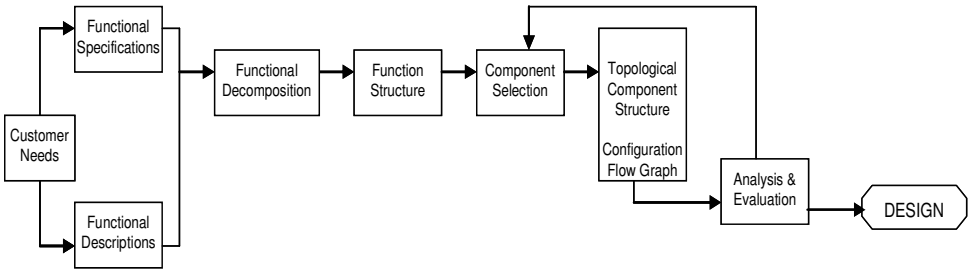


Figure 2.1 General flowchart of the computational synthesis process

The general flowchart of the developed computational synthesis process is shown in Figure 2.1. By following this process, a design is changed from an abstract set of customer needs to a conceptual configuration (Figures 1.2 - 1.5). The customer need analysis and the functional decomposition steps are performed by the designer. The computational design process is initiated at the level of a function structure (Pahl and Beitz, 1988; Otto and Wood, 2001). The output of the process is a conceptual design configuration where specific electromechanical components are first associated with individual or sets of sub-functions from the function structure and then composed into a design configuration based on their physical interactions. Feasibility and consistency is maintained throughout the design process while transitioning between these domains.

At the end, the design method manifests itself as a computational design tool. The major building blocks used during the development of this design tool can be visualized with the help of the architectural framework shown in Figure 2.2. A pyramid is a useful visualization here, since each level builds upon the foundation of the lower levels and reduces the complexity of its foundation in order to perform its intended tasks.

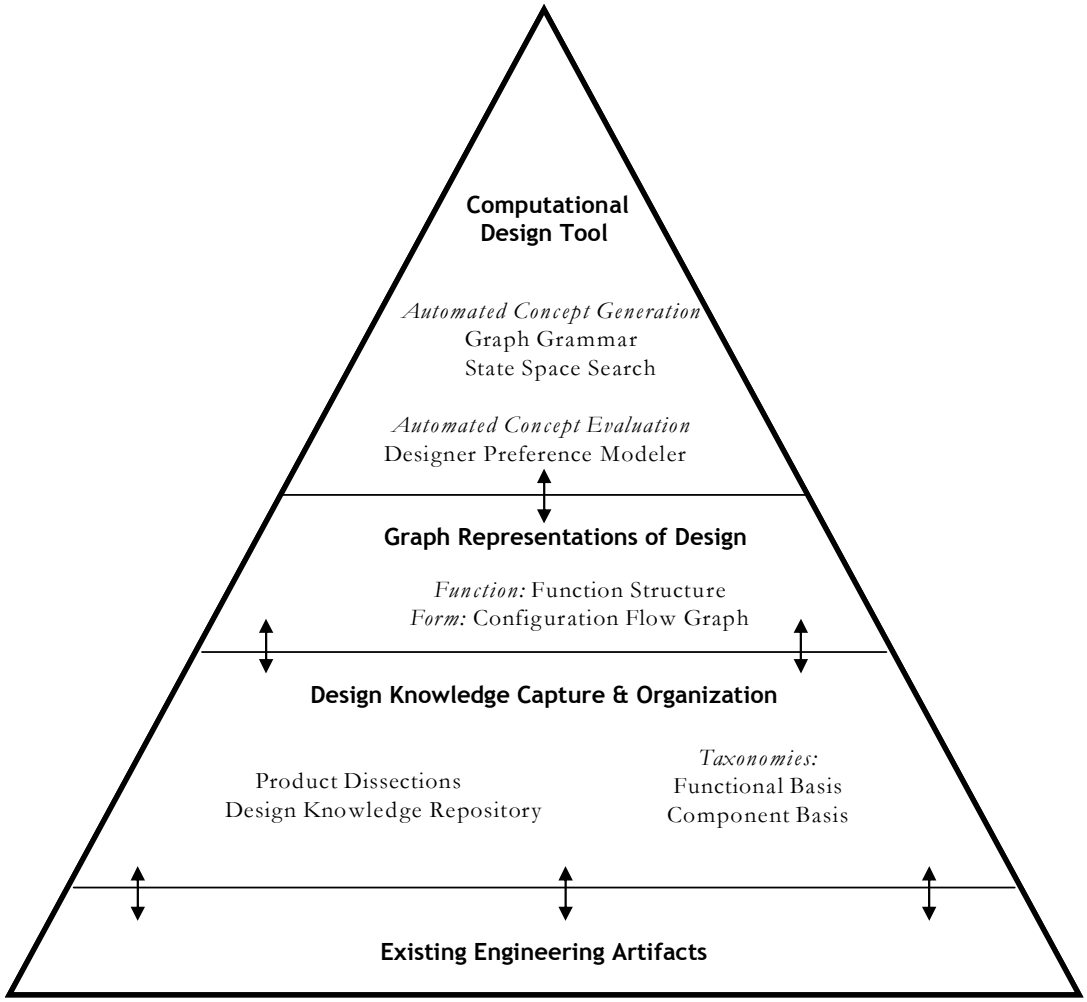


Figure 2.2 The architectural framework of the developed computational design tool

The primary motivation behind the developed method is to capture design knowledge from existing engineering artifacts, therefore the wealth of existing engineering artifacts constitute the bottom-most layer of the pyramid. The next layer is design knowledge capture and organization. At this level, design knowledge is gathered and organized using standardized taxonomies and a design knowledge repository. The middle layer is graph representations of design, where designs at two domains are represented in a graph-based format using the standardized taxonomies of the bottom layer. This layer hosts representations for design function (called function structures), and design form (called configuration flow graphs). These two graph representations constitute the foundation for the graph grammar language that the conceptual design tool is built upon. At the top layer, resides the design tool. It is at this level where concept variants are created and evaluated based on graph grammar and search algorithms developed. In the following chapters, each of the three main layers and their specifics are explained in detail.

Chapter 3: Design Knowledge Extraction and Organization

The fundamental research question addressed in this chapter is concerned with extracting and storing design information in such a way that the knowledge can be reused for future product design. The main hypothesis here is that valuable *design knowledge* is implicitly available in existing examples of engineered artifacts. Under this premise, design knowledge is extracted from existing products and stored for reuse in design knowledge databases and a web-based repository. Three main tasks at this step are studying existing products, recording and storing of design knowledge, and the development of standard taxonomies.

3.1 SYSTEMATIC PRODUCT DISSECTIONS AND EMPIRICAL PRODUCT ANALYSIS

The first step in this research is the dissection of consumer products. In this step, concrete experience with the existing product is emphasized. The aim is to fully understand the product in terms of functionality, components, product hierarchy, and configuration. Accordingly, design knowledge is extracted from a set of consumer products chosen for their low cost and wide variety. Each product is disassembled by strictly following the method presented in Otto & Wood (2001). To begin the product teardown, a disassembly plan is developed which lists each step that is to be taken. As the product is disassembled, a bill of materials (BOM) is constructed to document its components. Each component is labeled and photographed as it is removed from the product. The components that constitute assemblies or subassemblies are noted along with connectivity of components and their configurational relations. The resulting BOM is recorded in a tabular format and lists various component attributes. Figure 3.1 shows the teardown of a hand held vacuum cleaner at various stages and the BOM used for documenting its components.

Part	Qty	Description	Material (Color)	Function(s):Flow(s)	Mfg. Process
1.1	1	Nozzle	plastic (brown)	import: air, debris	injection molded
1.2	1	Rubber Flap Holder	plastic (translucent)	channel: debris, air	injection molded
1.3	1	Rubber Flap	rubber (black)	store debris	vulcanized
1.4	1	Filter Supporter	plastic (clear/white)	support bag	injection molded
1.5	1	Filter Bag	fibrous paper (white)	separate: debris from air	pressed
1.6	2	Staples	steel (silver)	secure: filter to supporter	rolled
2.1	2	Case Handle Half	plastic (brown)	transmit: weight; import: hand	injection molded
2.2	4	Screws	steel (shiny silvery)	hold case together	rolled
3(a).1	1	Release Button	plastic (brown)	export debris: clean filter	injection molded
3(a).2	1	On/Off Switch	plastic (brown)	actuate electricity	injection molded
3(b).1	1	Battery Cover	plastic (brown)	hold batteries in case	injection molded
4.1	1	Impeller	plastic (white)	convert mechanical energy to pneumatic energy	injection molded
4.2	1	Motor Support	plastic (white)	hold battery and motor	injection molded
4.3	2	Batteries	Ni-Cd type	store: electricity	mass produced
4.4	1	Motor	(dull silvery)	convert electricity to rotational energy	mass produced

Figure 3.1 The Bill of Materials of a handheld vacuum cleaner

3.2 DESIGN KNOWLEDGE REPOSITORY

The extracted component design information is organized and stored in a web-based design knowledge repository (http://function.basiceng.umr.edu/repository) that is managed at the University of Missouri-Rolla with contributions from the University of Texas at Austin, Pennsylvania State University, Bucknell University and the Virginia Polytechnic Institute and State University. The design data is recorded into the repository using an open source, cross-platform repository entry application. The entry template is

organized based on individual components in a product. Accordingly, information about each component is recorded on separate templates, which are then aggregated to construct a product level representation. There are three types of attributes captured for each component: descriptive attributes, physical attributes and functional attributes.

Examples of descriptive attributes are part number, quantity, part description, hierarchical (assembly) information, and predicted manufacturing processes. Physical attributes include weight, type of material, and component specific parametric information (examples include height, width, length, inside/outside diameters, thickness, etc). These attributes are physically measured for each component and recorded into the data template. Finally, the functional attributes include the function(s) each component fulfills, the flows (energy, material, and signal) that pass through each component, and the connectivity and resulting interfaces of components.

Because of the level of detail captured for the online database, the empirical dissection of products and the subsequent data entry into the repository is a very laborious process, typically ranging from 10-15 person-hours per product. These teardowns have been performed over the last three years through the efforts of several experienced graduate students, and the occasional undergraduate research assistants.

Following entry into the repository entry application, a platform-independent data set is output as XML and uploaded directly to the UMR Design Repository database. By creating Java Server Pages (JSP), the product data from the database server can be viewed as HTML through a standard web browser (Bohm et. al, 2004) as shown in Figure 3.2.

Given this format, design generation tools can be readily generated from single or multiple products and used in a variety of ways to enhance the design process (Bohm et.al, 2004). As one of these examples, this research leverages the repository data to

extract a design grammar language that captures the relationship between specific functions and solution principles that are used to fulfill them, which will be elaborated in Chapter 5.

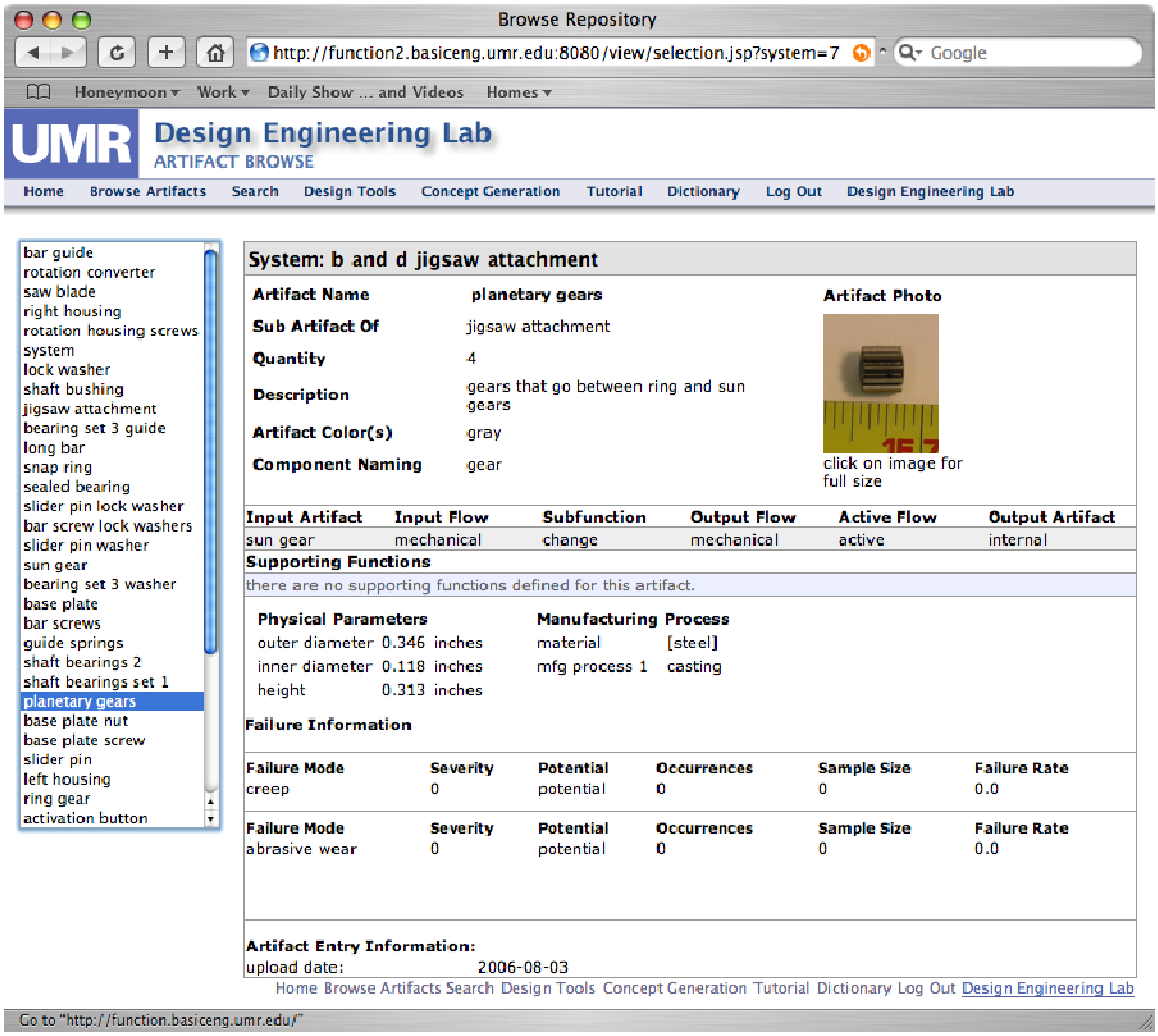


Figure 3.2 The UMR Design Repository web interface

3.2.1 Product Category

In this study, 23 consumer products are chosen to provide an empirical basis for the development of the design method and the resulting computational design tool. These

products offer simple technologies, yet the variety of the technologies used by the products cover a rich set of engineering solutions. These products are selected based on the fundamental flow types that govern their functioning. For example, an “iced tea maker” is primarily governed by liquid, and hydraulic energy flows, whereas a “vacuum cleaner” is governed by electrical and pneumatic energy flows. A “can opener”, on the other hand, is primarily a mechanical device that is governed by rotational and translational mechanical energy flows. In selecting the products that construct the empirical basis, attention has been given such that a comprehensive coverage of fundamental flow types (listed in Table 3.2 of Section 3.3.1) is attained. The complete list of products that are analyzed is shown in Table 3.1.

Table 3.1 The twenty-three products used as an empirical basis

Product List	
Power Screw Driver	Dishwasher
Can Opener	Eye Glass Cleaner
Hand Held Vacuum Cleaner	Electric Iron
Iced Tea Maker	Jar Opener
Presto Salad Shooter	Snow Cone Machine
Electric Knife	Traveling Sprinkler
Electric Toothbrush	Stir Chef
Electric Bug Vacuum	Hair Trimmer
Disposable Camera	Wine Opener
Knife Sharpener	Paper Shredder
Fruit Peeler	Electric Stapler
Pencil Sharpener	

3.2 TAXONOMIES FOR ORGANIZING DESIGN KNOWLEDGE

To derive uniformity and consistency in representing design knowledge, it is essential to develop standard languages. This section summarizes the standardized

vocabularies that are used to facilitate the development of the presented computational design tool.

There are three distinct knowledge representation schemes used as part of this research. These representations make use of two standard taxonomies: the *functional basis*, and the *component basis*. The former of is adopted from the literature (Hirtz et. al, 2002) and is presented as the means of both capturing the design knowledge and representing the conceptual design input required by the computational tool, whereas the latter is specifically developed for the purposes of this research as a classification system for component naming and for representing the conceptual design output generated by the computational tool.

3.3.1 Functional Basis

The Functional Basis is a set of function and flow terms that combine to form a sub-function description (in verb-object format). Shown in Tables 3.2 and 3.3, the hierarchically arranged basis terms, which are intended to span the entire electro-mechanical design space without repetition, are utilized during the generation of a black box model and functional model in order to encapsulate the actual or desired functionality of a product. In this approach, the designer follows a rigorous set of steps to define a new or redesigned product's functionality prior to exploring specific solutions for the design problem (Stone and Wood, 1999).

The black box model is constructed based on the overall product function and includes the various energy, material, and signal flows involved in the global functioning of the product. The detailed functional model is then derived from sub-functions that operate on the flows listed in the black box model. To briefly illustrate this technique, the functional model of an electric toothbrush was shown in Figure 1.5. Repeatability, ease in storing and sharing design information, and increased scope in the search for solutions

are some of the advantages these functional models exhibit (Stone and Wood, 1999; Kurfman et al., 2003)

Table 3.2 Flow classes and their basic categorizations

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

Table 3.3 Function classes and their basic categorizations

	Primary Class	Secondary Class
Functional Basis Reconciled Function Set	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

3.3.2 Component Basis

A critical challenge for the presented research was to determine a systematic means of cataloging and classifying component design knowledge. Considerable effort has been put forward to develop a component taxonomy that would define a hierarchical

list of distinct component types and their definitions. In developing this taxonomy, a perspective is taken that promotes *functionality* as the principle classification scheme for a component. Accordingly, each component is classified under a specific component type according to a distinct function-based definition, as illustrated in the excerpt shown in Table 3.4.

Table 3.4 An excerpt of component terms and definitions

Primary Component Classification	Secondary Component Classification	Component Term	Component Subset	Synonyms	Definition
Branchers	Separaters	...			
	Distributors	...			
Channelers	Importers/Exporters	...			
	Transferors	Carousel			A device used to move material in a continuous circular path.
		Conveyor			A device used to move material in a linear path.
		Electric Conductor		*lead*	A device used to transmit electrical energy from one component to another.
			Electric Wire		An electric conductor in the form of a thin, flexible thread or rod.
			Electric Plate		An electric conductor in the form of a thin, flat sheet or strip.
		Electric Socket			A device in the form of a receptacle that transmits electrical energy via a detachable connection with an electric plug.
		Electric Plug			A device in the form of a plug that transmits electrical energy via a detachable connection with an electric socket.
		Belt		*strap, girdle, band, restraint*	A device shaped as an endless loop of flexible material between two rotating shafts or pulleys used to transmit mechanical energy.
		...			
	Guiders	Hinge		*pivot, axis, pin, hold down, jam, post, peg, dowel*	A device that allows rigidly connected materials to rotate relative to each other about an axis, such as the revolution of a lid, valve, gate or door, etc.
		Diode			A semiconductor device which allows current to flow in only one direction.
		...			
Connectors	Couplers				
	Mixers				
...	...	...			

The approach taken here is similar to the development of ontologies. The concept of ontology, as used in knowledge engineering, is described as (Uschold and Gruninger, 1996) "a term used to refer to the shared understanding of some domain interest." Neches, et al. (1991) state: "An ontology defines the basic terms and relations compromising the vocabulary of a topic area." Several ontologies have been proposed in the engineering design literature (Liang and Paredis, 2004; Hovarth et al., 1994; 1998; Kitamura, 2003; Li et al., 2005; Stahovich et al., 1993). Liang et al. (2004) suggests a port ontology as a tool for conceptual modeling and generating system architecture. Stahovich, et al. (1993) proposed that the fundamental ontology for mechanical devices

should be based on physical behavior, not structure. This research attempts to determine the behavior of a mechanical device from a description of its structure and its deriving inputs. Hovart, et al. (1998) have defined a general ontology to model design concepts and the interactions between them. The cataloged concepts are then initiated in a computer based functional synthesis process. Similar to this approach, Kitamura and Mizoguchi (2003) have proposed an ontology to describe functionality of physical artifacts and its relationship with behavior based on a language called FBRL (function behavior representation language).

In this research, a taxonomy for describing the electromechanical component space is defined. The presented taxonomy complements the functional basis representation (Hirtz et al., 2002) and allows for well-defined function-based types of components to be used in the creation of required design representations and the computational design tool that results. It also eliminates artifact redundancies that may not be immediately evident due to variations in user-dependent artifact naming. For example, separate components under different products may be named "motor 1", "shaded pole induction motor", or "dc motor". Using the component taxonomy, each of these artifacts would be identified as similar and tagged as an "electric motor".

3.2.2.1 The Classification Hierarchy

The goal of classification in this research is focused more on the practical use of the proposed hierarchical taxonomy. For this reason, the hierarchical framework was initially established from the notion that device function is an integral and critical characteristic of a component from the perspective of concept selection during the design process. As a starting point, the list of primary and secondary level function terms from the Functional Basis (Hirtz et al., 2002), shown in Tables 3.2 and 3.3, were used to designate the primary and secondary levels of the component framework. In order to

successfully place component terms into the taxonomy, the functional traits of each component is established by modeling the components by defining appropriate input-output ports through which they connect to other components. A rigorous process is then followed to establish the hierarchy and to place component types within the hierarchy. (Kurtoglu et al., 2005).

The individual component terms and associated definitions represent the different types of components. The definitions of these terms are critical to the usefulness of the proposed taxonomy. In defining these component terms, disagreements may exist over how narrowly to define different terms. For example, should an axle and a drive shaft be classified under the same component term? Should a flexible hose be classified under a different component term than a rigid tube? In the case of the axle and drive shaft, these two components solve different functionality and would, therefore, be placed under different branches of the proposed taxonomy. The flexible hose and rigid tube are functionally similar, so a decision must be made about whether to group them together under a broad definition or separate them into more specific groups. When defining terms, effort was made to determine whether a new (separate) definition would be beneficial from the perspective of a designer in the early conceptual stages of design, for example deciding whether to use a flexible versus a rigid tube to transfer a material would be less useful when initially generating concepts than deciding whether to use a tube versus a conveyor. To help evaluate whether terms were defined at a low enough level of detail while preserving necessary coverage of the component space, additional consideration was made and two criteria were defined: *completeness* and *exclusivity*.

Completeness: The measure of how well a list of components captures the complete set of all electromechanical components.

Exclusivity: The measure of the independence of terms in the taxonomy. It is the opposite of overlap or redundancy of taxonomy terms.

These two criteria are enforced at the component term and component subset levels of the hierarchy (see Table 3.4 for an excerpt). The completeness criterion motivates one to construct a taxonomy that covers all the concepts that are relevant in the electromechanical component domain. In general, completeness within a domain is accomplished automatically with an empirical study of existing products. It is likely that with each additional product that is dissected and catalogued in the repository, the return in the number of newly defined fundamental component concepts begins to reduce. Of course, a large number of artifacts will need to be dissected before a change in the rate of new concepts is noticeably decreased.

The challenge in such research is to heed the exclusivity criterion. Technically, there could be as many component types as the number of components that exist in all electromechanical devices. This, of course, is impractical, and does not take advantage of component types that are well accepted (e.g., gears). As briefly described before, the key in establishing the exclusivity criterion is to carefully define what constitutes a new fundamental component type.

To achieve this, two guidelines are followed: (1) any fundamental component type should be as general as possible but specific enough to allow the user to build a clear abstraction of a component which can be used during conceptual design. (2) The defined taxonomy as a whole should include a sufficient number of component terms (i.e. building blocks) to allow the user to represent a variety of concepts. This is illustrated in Figure 3.3. In Figure 3.3a, two artifacts are shown that “convert electrical energy to mechanical energy”. The fact that they differ in their size and shape does not deter us from conceding that they are actually two instantiations of the same component concept,

which we refer to as “an electric motor”. Therefore, they are represented under the same fundamental component term: electric motor. Figure 3.3b, on the other hand, shows two fasteners. These two artifacts not only share the same functionality of “coupling two objects”, but also look very similar. Yet, they are instantiations of two different component concepts, one of which is referred to as “a screw” and the other as “a nut and bolt”. Simply representing both under a single class named “fastener” results in a component type that may be too generic. This contradicts with the first guideline presented for establishing the exclusivity criterion. Therefore, the two artifacts shown in Figure 3.3 are granted their own fundamental component terms and are grouped under a functionally more abstract “fastener” class among other widely accepted fastener types such as solder, rivets, and welds. By following these two guidelines, the collaborative efforts of defining an exclusive set of component concepts led to a total of 135 component terms, and the complete list is shown in Appendix A.

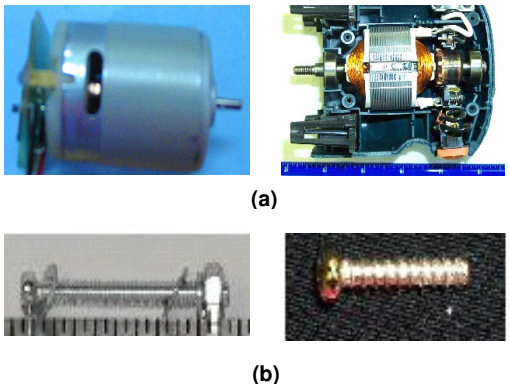


Figure 3.3 Instantiations of (a) two “motors” and (b) two “fasteners”

In the next few chapters, the details of the computational design application will be presented which relies on the developed component taxonomy in order to automate the concept generation phase of the design process.

Chapter 4: Representations of Design

One of the main difficulties in supporting conceptual design is the complexity involved in modeling and representing a design solution (Hsu and Woon, 1998). While there are many formal techniques that have been developed to represent different aspects of a design solution, two groups of representations are of particular importance to the conceptual phase of design: *representation of function*, and *representation of form.*

This importance is emphasized in a certain view of design that defines the process of concept generation as the transformation from function to form, or more specifically the creation of a form that meets functional requirements (Otto and Wood, 2001). Influenced by this view of design, this chapter provides an overview of the representation of function and form in general, and the specifics of the graph-based representations used by the computational grammar in particular.

4.1 REVIEW OF FUNCTION-BASED REPRESENTATIONS

Function based representations allow designers to represent solutions independent of their form. The most common approach to functional representation is to decompose the system into sub-functions to yield a functional graph that is used to seek solutions to sub-functions. The form of the system then follows from the assembly of all sub-function solutions. Most notable of these approaches is the function structures of Pahl and Beitz (1988) that represents the German school of design. Similar methods include Hundal (1990) who formulates six function classes complete with more specific functions in each class, though he does not claim to have an exhaustive list of mechanical design functions. Another approach uses the 20 subsystem representations from living systems theory to represent mechanical design functions (Koch, et al., 1994; Malmqvist, et al., 1996) The Soviet Union era design methodology known as the Theory of Inventive Problem Solving

(TIPS) uses a set of 30 functional descriptions to describe all mechanical design functions (Altshuller, 1984). Kirschman and Fadel (1998) propose four basic mechanical functions groups, but vary from the standard verb-object sub-function description common to most methodologies. This work appears to be the first attempt at creating a common vocabulary of design that leads to common functional models of products. Other researches have also pursued a standard functional design language (Little, et al., 1997; Otto and Wood, 1997; Stone and Wood, 1999; Murdock, et al., 1997; Szykman, et al., 1999; Hirtz, et al., 2002). The result of these efforts is the Functional Basis design language presented in Section 3.3.1. In functional basis, engineering functions are categorized as 8 classes that are further specified as basic categories. The NIST design repository project (Szykman, et al., 1999 & 2002; NIST, 2000) is another framework for storing and representing functional design information along with four other major classes. Japanese researchers have also explored a consistent language for describing the functionality of products and relating it to product behavior (Kitamura and Mizoguchi, 1998 & 1999; Umeda and Tomiyama, 1997; Sasajima, et al., 1995).

Another example of a behavior oriented approach to function modeling is the bond graph formalism (Paynter, 1961, Karnopp and Rosenberg, 1975). The bond graph formalism represents a dynamic system as a composition of components where each component deals with power flow and has effort parameters (such as pressure, voltage, and force) and flow parameters (such as flow rate, current, and velocity). The main advantage of bond graph technique is the fact that it can model a variety of domains including mechanical, electrical, and hydrodynamic systems by using a generic, fundamental language of dynamic operators. While bond graphs originated as an analysis method, various projects (Bracewell and Sharpe, 1996; Finger and Riderle, 1989) have since realized the potential of bond graphs as a foundation for design synthesis. The

design platform proposed by Bracewell and Sharpe is called Schemebuilder. It uses function-means trees to hierarchically represent required functionality of a system. Schemebuilder incorporates a component library with the function-means representation to build a functional-embodiment knowledge base which is used to instantiate components to address certain functions. By connecting component models, a bond-graph model of a design solution can be generated which is later used for behavioral simulation and performance assessment. The main limitation of the Schemebuilder is its scope that only deals with functions represented as power flows.

On the other hand, several researchers have developed high-level design languages to describe product function. The Causal Function Representation Language (CFRL) defined function with a triplet $\{D_F, C_F, G_F\}$, in which D_F denotes the part function, C_F is the context where the part is to function, and G_F indicates the description of the functional goal to be accomplished (Iwasaki et al., 1993). This method allows the designer to determine the functional specification and physical structure under a unified framework. CFRL models expected behavior of a device using logic based propositions that capture causal sequence of events required for proper functioning, and thereby facilitates analysis of designs through a model-based qualitative simulation that compares predicted behavior against system function. Despite its well-defined formalism, CFRL is limited in its implementation to only small-scale examples. Similarly, Sturges et al. (1996) used a block diagram approach based on function logic. To describe complex systems, this representation schema includes mathematical relationship equations that govern functional blocks.

Gero et al. presented a formalism called FBS (Qian and Gero, 1996) that defines structure, behavior, and function of a system. In this technique, relations among function, behavior, structure are utilized to extract paths in a FBS-graph which are then used to

retrieve design information to conduct analogy-based design. Similarly, the Structure-Behavior-Function model (Bhatta et al., 1994) is a system that captures function and has a design-case memory that represents each case as an SBF model for analogy-based design. The Function-Behavior-State modeler (Umeda et al., 1996) is another computer-based implementation that is developed to model function and physical behavior of components. FBS modeler allows more flexible definition of device function and thus is applicable to a variety of domains.

Other functional modeling approaches include Design Structure Matrices (DSM) that are used for task decomposition and design reviews, and matrix-based representations that capture customer needs and function at a product level (McAdams et al., 1999; Stone et al., 2000) to facilitate a computational approach to design by analogy.

4.2 REVIEW OF FORM-BASED REPRESENTATIONS

Most conceptual design techniques include representations of the final form of the design object. These representations take a variety of forms changing from physical topology of a conceptual solution to representations of 2-D, and 3-D geometric shapes.

Popular representations of geometric shapes include Constructive Solid Geometry (CSG), Boundary Representations (B-Rep), and feature-based representations (Mortenson, 1997). The CSG approach models 3-D geometry using a set of primitives such as cubes, pyramids, or cylinders. Complex shapes can be built from these primitives through a set of Boolean operations. In B-Rep approaches, a shape is represented in terms of its boundary information such as faces, edges, and vertices and the topological relations among them. Finally, the feature-based representations represent objects using its constituent features including holes, slots, etc.

These three geometrical representations have been implemented into many commercial CAD systems. Although they have been widely used for many years for

representing and manipulating geometric form of objects, their use is highly limited during conceptual design, where the geometry of components is still ill-defined. Moreover, these representations lack a functional description of components thus impeding computational synthesis and analysis of a conceptual structure or configuration.

On the other hand, in the recent years, approaches are emerging that capture the interaction between the function and geometric form of a design such that the representations can be used by synthesis oriented computational tools. Welch and Dixon (1994) developed behavior graphs that are based on research in qualitative physics. This representation explicitly defines physical connectivity of design elements using functional parameters and embodiment. Campbell (2000) extended this representation to facilitate the development of an agent-based synthesis method known as A-Design.

Functional grammars (Schmidt et al., 1995; Campbell, 2007) and shape grammars (Stiny, 1980) provide another representation to represent form of a design object. These representations vary from capturing simple topology of components to representing basic properties of shape of an object. This family of representations is elaborated in the next chapter.

4.3 REPRESENTATION OF THE DESIGN SPACE FOR GRAMMAR DEVELOPMENT

Following the view of design that defines the process of concept generation as the transformation from function to form, this research develops a computational method that converts a function-based model of a design solution to a configuration-based model. These models within the computational method are represented using two graph-based representations: *function structures* and *configuration flow graphs*. The former of these captures the design function, whereas the latter represents the configuration or topology of a design. These two graph representations constitute the foundation for the graph

grammar language that the conceptual design tool is built upon. They are discussed in the following sections, but first a brief overview of graph based representations is provided.

4.3.1 Graphs as a Representation Scheme

A graph is a collection of nodes interconnected by arcs. Mathematically, a graph G = (V, E), is composed of V, a set of nodes and E, a set of arcs connecting the nodes in V. An arc E = (u, w) links to a pair of nodes u and w. Graphs find application is various types of engineering systems ranging from electronic circuits to networks of roads, communication and scheduling.

One of the main advantages of using graphs to model different aspects of a design is that graph theory is a well-established field of study. By using graph-based models, one can leverage resources of the many existing graph manipulation algorithms with sound theoretical bases. Moreover, graphs are considered to be superior to traditional parametric representations of design especially for computations at the conceptual phases, where interconnectivity of design elements is more important than their parametric details. Because of these advantages, "graphs" are selected as the underlying modeling scheme in this research.

4.3.2 Representation of Function: Function Structure

A function structure (Pahl and Beitz, 1988) is a graphical representation of the decomposition of the overall function of a product into smaller, more elemental sub-functions. The sub-functions are connected by flows that are of type energy, material and signal. Overall, a function structure represents the transformation of input flows into output flows at the system level. Otto and Wood (2001) and Kurfman et al. (2001) put forth a method to build a function structure starting from the customer needs. Obtaining the customer needs, the generation of a black box model, the creation of function chains

for each input flow, and the aggregation of function chains into a function structure are the sequence of steps that lead to the construction of a function structure. This process is illustrated in Figure 4.1.

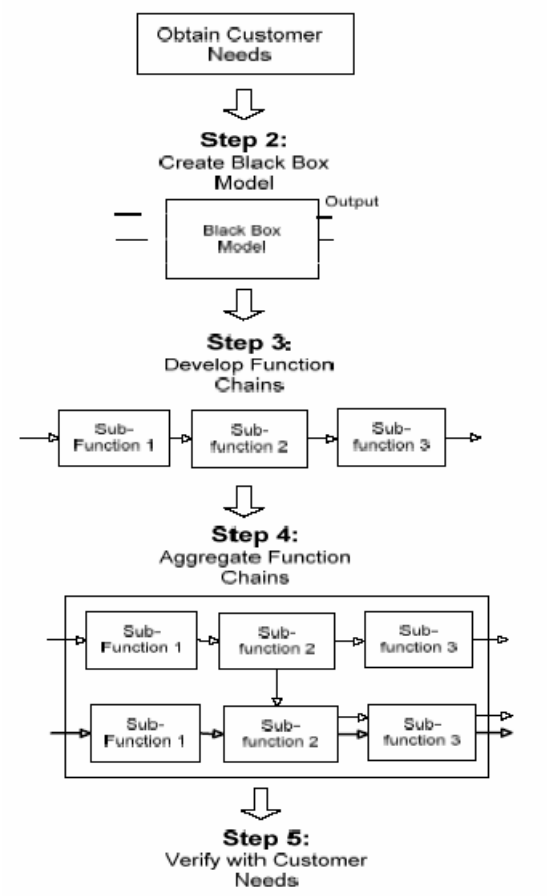


Figure 4.1 Function structure derivation steps (Kurfman, 2001)

In order to attain a repeatable formation of function structures, the aforementioned functional basis (Hirtz et al., 2002) is used as a standard vocabulary during the construction of function structures. To illustrate examples, the function structures of an electric toothbrush and a traveling sprinkler are presented in Figures 4.2 and 4.3.

4.3.3 Representation of Form: Configuration Flow Graph

This research attempts to generate solutions to conceptual design problems. In such problems, it is essential to determine an optimal configuration of components prior to dealing with parametric details of component geometry, or shape. Naturally, there is need to represent topologies of conceptual design solutions that capture "form" without detailed geometric information. The representation scheme presented in this section, called the configuration flow graph (CFG), is developed specifically to address this need. It is an improvement over existing examples of topological connectedness graphs (Welch and Dixon, 1994; Bracewell and Sharpe,1996; Campbell, 2000) in that it's scope is not limited to those of standard bond-graph elements which can only represent components associated with power flows. The development of configuration flow graphs is motivated by the need to provide a broader set of fundamental components, and interaction types that are not limited to power transformation.

A Configuration Flow Graph (CFG) is a graphical representation of how components in a design are connected. In a CFG, nodes represent product components, and arcs represent energy, material or signal flows between them. For flow naming the functional basis terminology (Hirtz et al., 2002) is adopted, while the component types used are selected from the component terms of the developed component basis (Kurtoglu, et al., 2005). At an abstract level, the CFG also represents the behavior of components by modeling them as "black box" entities that transform certain input flows into certain output flows. From this perspective, the behavioral representation is similar to port based methodologies such as those presented in (Paredis et al., 2001; Liang et al., 2004)

The CFG is a specific implementation of what some loosely define as the topology, the architecture, or the configuration of a product. The graph is also similar to an exploded view in that components (often drawn isometrically) are shown connected to

one another through arcs or assembly paths. Figures 4.4 and 4.5 show examples of CFG's for an electric toothbrush and a traveling sprinkler accompanied with associated conceptual sketches. As can be seen from the figures, components that are present in a design, their connectivity, and physical interfaces between a design's components can be captured using a configuration flow graph.

The use of the configuration flow graphs as a modeling tool is not limited to any specific category of designs. Similar to how function structures are used to functionally model any product, configuration flow graphs can be utilized in modeling the configuration, or architecture of any electromechanical device. To illustrate this, Figures 4.6 and 4.7 show two common architectures of hybrid vehicles represented using configuration flow graphs.

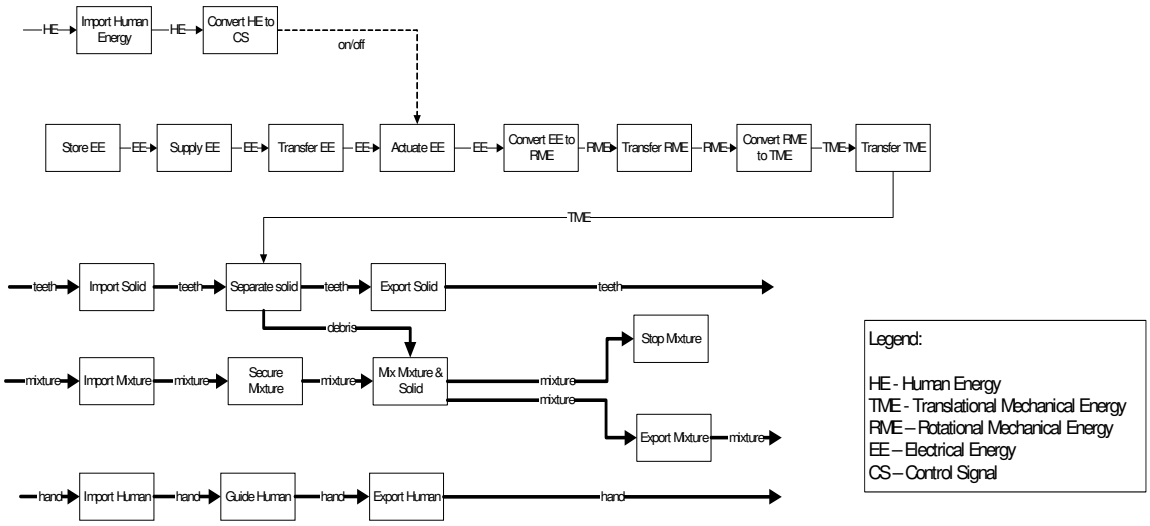


Figure 4.2 Function structure of an electric toothbrush

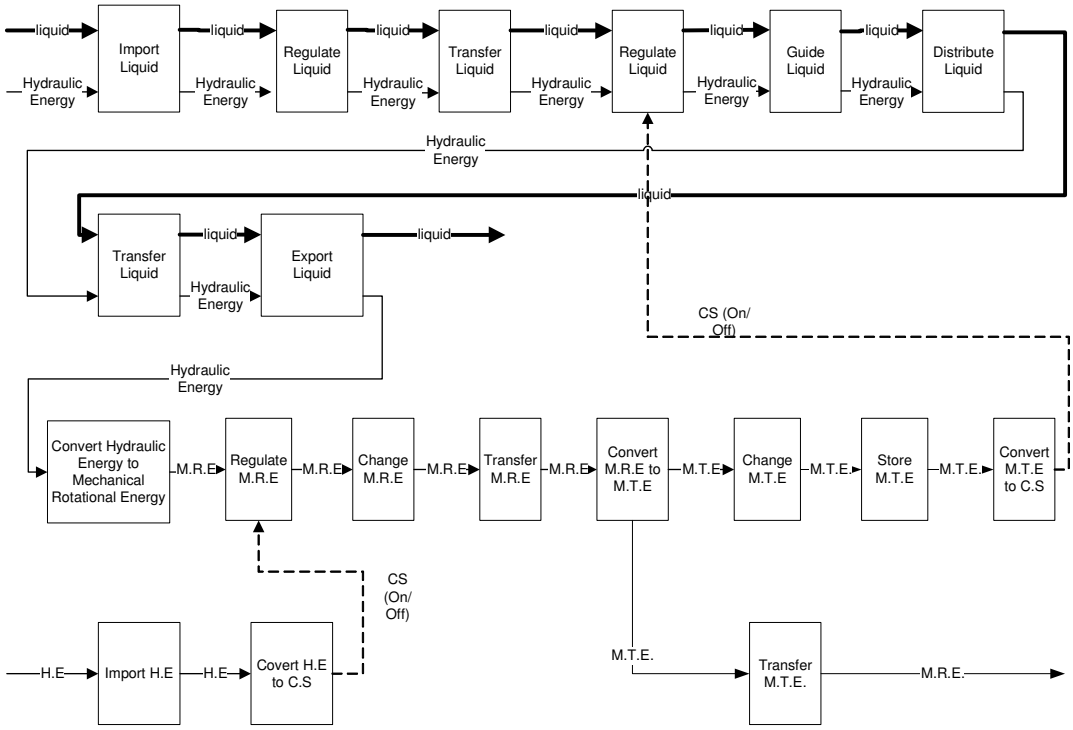


Figure 4.3 Function structure of a traveling sprinkler

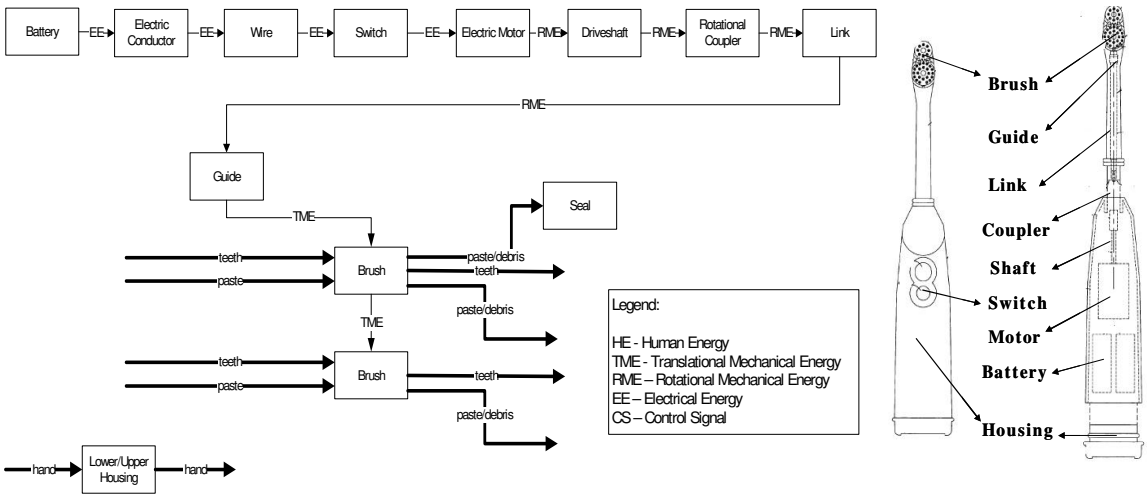


Figure 4.4 Configuration flow graph of an electric toothbrush

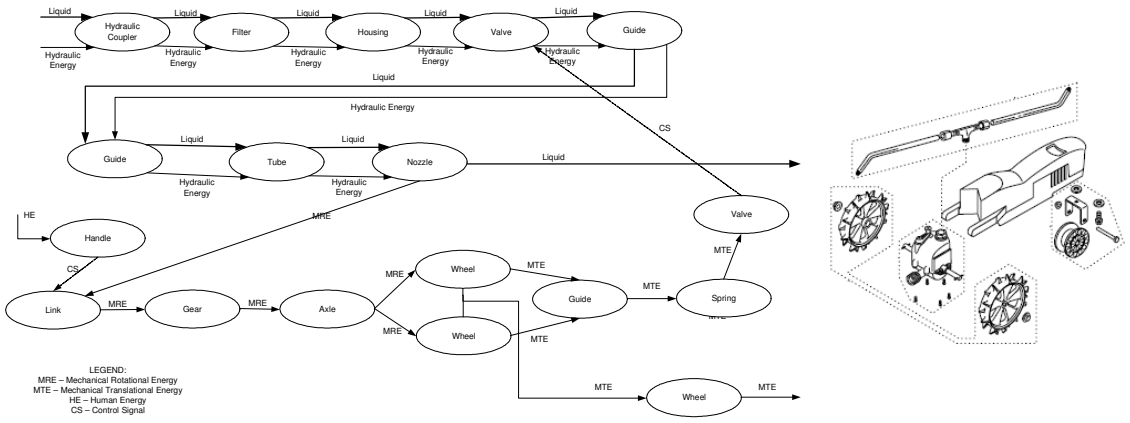


Figure 4.5 Configuration flow graph of a traveling sprinkler (Figure of exploded view from Nelson RainTrain® 1865 Assembly Manual)

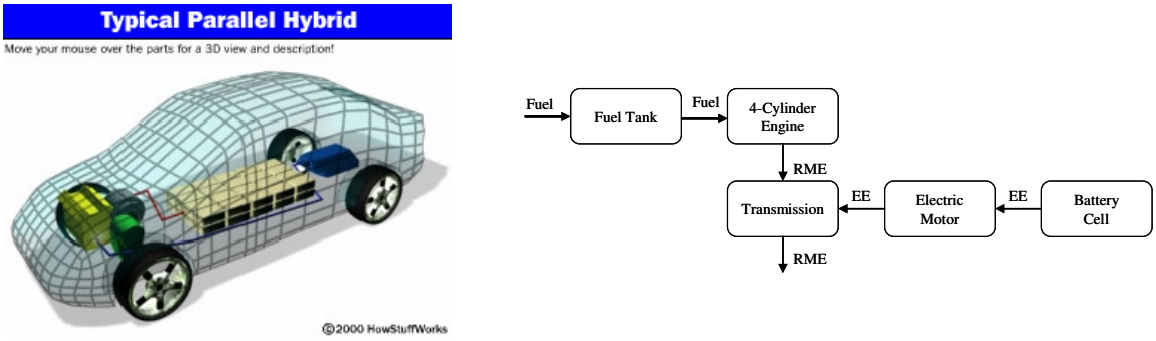


Figure 4.6 Configuration flow graph of a parallel hybrid car architecture (figure from howstuffworks.com)

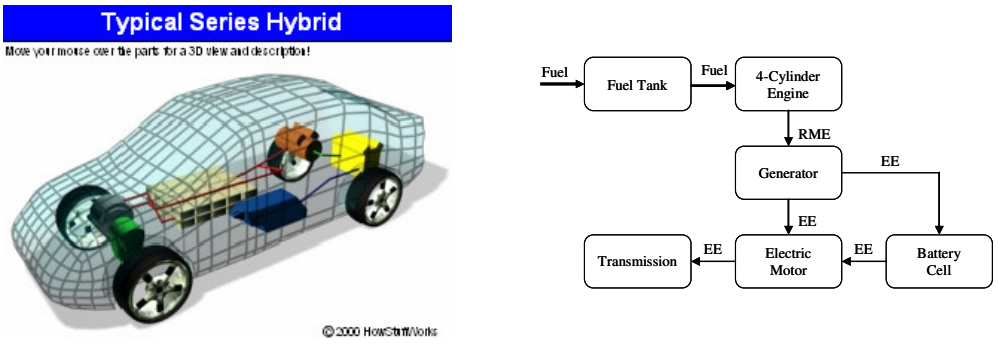


Figure 4.7 Configuration flow graph of a series hybrid car architecture (figure from howstuffworks.com)

Chapter 5: Development of a Grammar for Conceptual Design

Graph transformation systems, or graph grammars, reside in graph theory research as a way to rigorously define mathematical operations such as addition and intersection of graphs. Recently, engineering design researchers have discovered that graph grammars provide a flexible yet ideally structured approach to the creation of complex engineering systems.

These approaches capture the transitions or the production rules for creating a solution, as opposed to storing the solutions themselves. Accordingly, an artifact's development from its inception to its final configuration is considered as a series of modifications. The initial specification can be represented as a simple graph in which the desired inputs and outputs are cast as arcs and nodes of the to-be-designed artifact. From this initial specification, the design process can be viewed as a progression of graph transformations that lead to the final configuration (Campbell, 2007). This interpretation of the design process makes graph grammars very suitable for computationally modeling the open-ended nature of conceptual design, where designers explore various ideas, decisions, and modifications to previous designs to arrive at feasible solutions.

This chapter focuses on the development of a graph grammar for conceptual design. A brief overview of the development and application of shape and graph grammars is provided. This is followed by a discussion of the fundamentals of graph grammars, grammar derivation, and the description of the process used in developing the grammar rules for conceptual design. Finally, the resulting rule set is presented.

5.1 Review of Shape and Graph Grammars

A shape grammar is a set of shape rules that apply in a step-by-step way to generate a set, or language, of designs. The rules of a shape grammar generate or

compute designs, and the rules themselves are descriptions of the forms of generated designs. It was Stiny (1980) who first introduced and used shape grammars in architectural research. According to Stiny and Gips (1980), shape grammars can be thought of as a type of expert or production system based on geometry. The application of shape grammars extended to spatial grammars and algorithms are presented for the maximal representation of a shape. (Krishnamurti, 1992; Krishnamurti et al., 1993). Other architectural examples include shape grammars that are constructed to capture the style of a specific period, as was the case with Cagdas's grammar (1996) describing traditional Turkish houses, or the style of a specific designer such as Koning's grammar (1981) of Frank Lloyd Wright houses.

Grammars are also widely used in engineering applications. Agarwal and Cagan's coffee maker grammar (1998) was one of first examples of using grammars for product design. Their grammar described a language that generates a large class of coffee makers. Shea et al. (1997) presented a parametric shape grammar for the design of truss structures that uses recursive annealing techniques for topology optimization. Other engineering applications include Brown, et al. (1997), who presented a lathe grammar, Schmidt and Cagan's grammar for machine design (1995), Starling and Shea's grammars for mechanical clocks (2003) and gear trains (2005).

One of the interesting implementations of grammars to function-based design is the work of Sridharan and Campbell (2004). This research showed how a set of 69 grammar rules are developed to guide the design process from an initial functional goal to a detailed function structure. The presented work builds upon this research and significantly extends the graph grammar approach to function-based design. Accordingly, it starts with a function structure and seeks multiple configuration solutions for the various sub-functions of the function structure. To achieve this, it integrates the formal

function based synthesis method (Pahl and Beitz, 1988) with the configuration flow graph representation developed to facilitate the formulation of a graph grammar language that captures the knowledge employed by designers in mapping function to form during conceptual design.

5.2 FUNDAMENTALS OF GRAPH GRAMMARS

Graph grammars are comprised of *rules* for manipulating nodes and arcs within a graph. The rules create a formal language for generating and updating complex designs from an initial graph-based specification. The development of the rules encapsulate a set a valid operations that can occur in the development of a design. Through the application of each grammar rule the design is transformed into a new state, incrementally evolving towards a desired solution. The rules are established prior to the design process and capture a certain type of design knowledge that is inherent to the problem. Moreover, the rules can be formulated in such a way that the final solution meets the constraints of the problem. The knowledge captured in the rules offer the option of exploring the design alternatives as well as automating the design generation process.

A typical graph grammar rule is compromised of a left-hand side (LHS) and a right-hand side (RHS). The LHS contains the conditions, upon which the applicability of a rule is determined. Accordingly, the LHS describes the state of the graph for a particular rule to be applicable. The RHS, on the other hand, contains the resulting graph transformation. It describes the new state of the graph after the application of the rule. An example of graph transformations using a grammar rule is shown in Figure 5.1. The LHS of the rule specifies the application condition which states that the rule is applicable at a location of the 'host graph' where there is a circle node, and a triangle node both connected to a square node. As shown in the figure, there are two locations within the host graph where this rule is applicable. The RHS of the rule specifies the graph

transformation that the host graph should undergo. According to the RHS, the triangle and the circle nodes should be removed from the host graph and replaced by the addition of a new cross node. The figure shows the resulting host graphs after the rule is applied at two different locations.

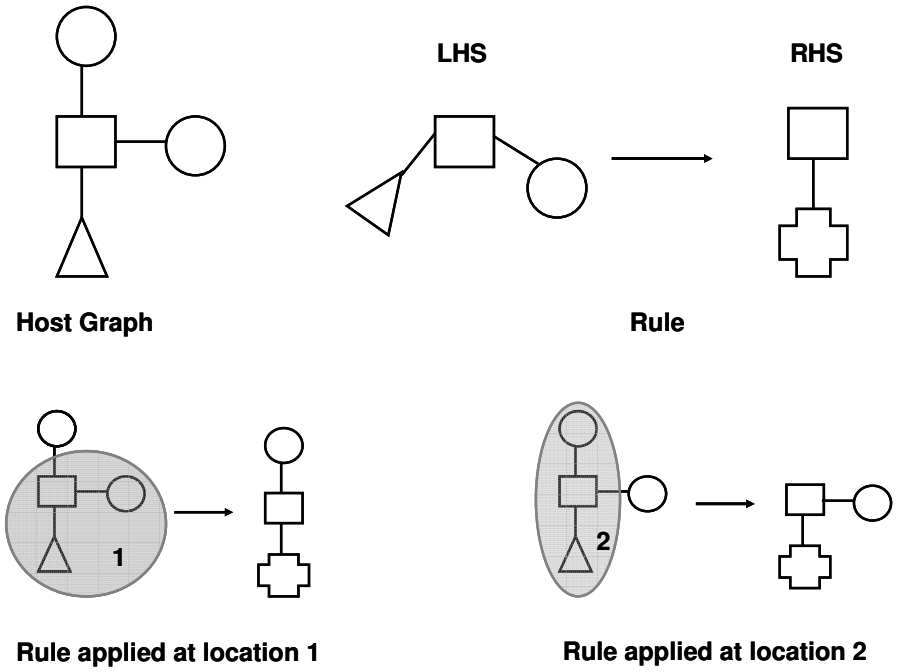


Figure 5.1 An example of graph transformation using rules

There are various algebraic methods developed that define how the graph transformations should be carried out. In this research, the graph transformations are implemented using the double-push out method (Rosenberg, 1997). The details of this method are beyond the scope of this dissertation and can be found in (Campbell, 2007).

The usage of a design grammar helps to generate a wide range of solutions by altering the way the rules are applied. The grammar affords a representation of the design space as a tree of solutions built from an initial specification. Each transition in the tree

corresponds to an application of a rule, thus incrementally building a final solution which is represented as one of the leaves of the tree. This process is illustrated in Figure 5.2 for a grammar used for designing electric circuits. As is evident from the tree, the result of rule applications generates a design space that requires navigation techniques to enable search for a desired or optimal solution. The issue of implementation of the grammar then becomes one of controlled searches through this space of solutions.

The search process gives the designer the potential to explore a large number of alternative designs which include many alternatives that might have been overlooked without the aid of a grammar, thus paving the way for possible innovative designs.

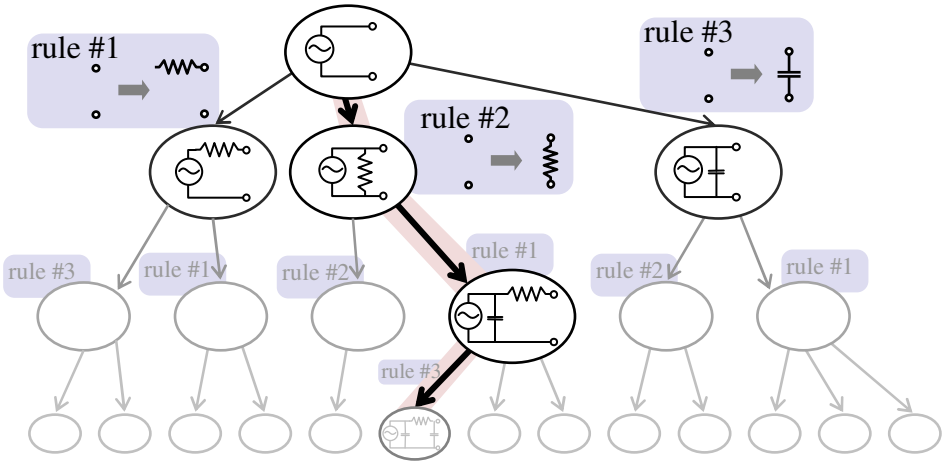


Figure 5.2 A visualization of a search tree in building a design solution from an initial graph using a grammar rule set. (Campbell, 2007)

5.3 DERIVATION OF A GRAMMAR FOR CONCEPTUAL DESIGN

In this section, the derivation of the graph grammar for conceptual design is presented. The objective of the implementation of the conceptual design grammar is to create new design configurations that address a given set of functional design requirements. To achieve this goal, the information extracted through empirical product

analyses is formulated as “grammar rules” that capture the knowledge of the original designer which is employed in the construction of conceptual solutions. Specifically, the derived graph grammar language encodes the mapping from the functional space to the configuration space.

The first step in applying any graph grammar is to represent the problem in the form of a graph. As previously discussed, a design solution in this research is represented using function structures and configuration flow graphs. Through systematic product teardowns and data gathering in the repository, an existing product’s CFG and FS are captured. Then, the mapping between these two graphs is extracted. By examining these relationships, grammar rules are carefully constructed that capture the ways in which components fulfill various functions.

When implemented the developed grammar constructs a configuration flow graph given the function structure for a conceptual design. In this regard, the approach is a novel grammar application in that a new graph (CFG) is constructed based on a separate, initially completed graph (i.e. the function structure). To perform this graph transformation, the grammar rules are defined to add components to the CFG that maintain a valid connection of components as well as meet specific function requirements specified with the function structure. Each of the rules developed are modeled after basic grammar conventions where rules are comprised of a left-hand side (LHS) and right-hand side (RHS). The left-hand side contains the state that must be recognized in the function structure and the right-hand side shows how the design is updated to a new configuration. Consider Figure 5.3 as an example. This rule states that if functions 1,2,3 and flows 1 through 5 are recognized in the host graph (i.e the function structure) then the design can be updated by the addition of components 1 and 2 and their physical interactions represented by flows 1 through 4. Note that the same right-hand side could have multiple

left-hand side alternatives. These cases are represented as separate rules and account for function sharing. Similarly, the same left-hand side could have multiple right-hand sides defined. This ensures enumeration of different component alternatives for a given sub-function or set of sub-functions.

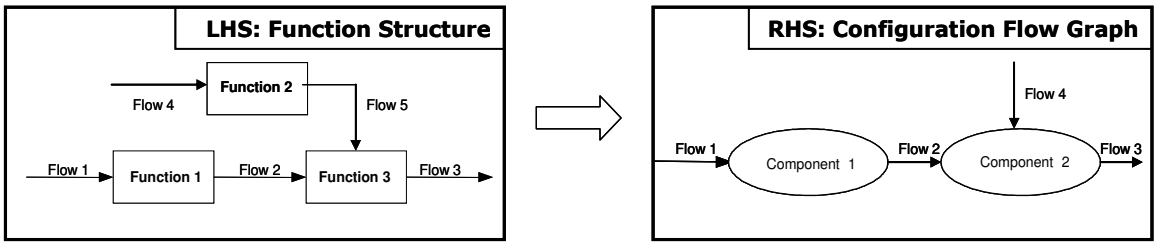


Figure 5.3 A hypothetical grammar rule

In reality, each rule represents a *design decision* that shows how a functional requirement was transformed into an embodied solution in an actual design. In order to extract these mappings in a consistent manner, the flow information provided by the two graphs is utilized. Advantage is taken of the fact that the two graphs contain the same flow types. This allows one to "follow the flow paths" in both graphs, and to define strict boundaries that isolate the mapping between functional nodes of a function structure and component nodes of a configuration flow graph. This procedure is illustrated schematically in Figure 5.4, where design rules derived from an empirical analysis of an electric toothbrush product are shown. The grammar rules define what components can be used in creating a new design in order to meet certain functional requirements specified by the function structure of the new design. For example, the first rule states that if functions "store electrical energy" and "supply electrical energy" are required in a design problem, then a "battery" can be used to address the specified functional need. Similarly, the second rule states that a "driveshaft" and a "rotational coupler" pair can be used to fulfill the functional requirement of "transfer rotational mechanical energy".

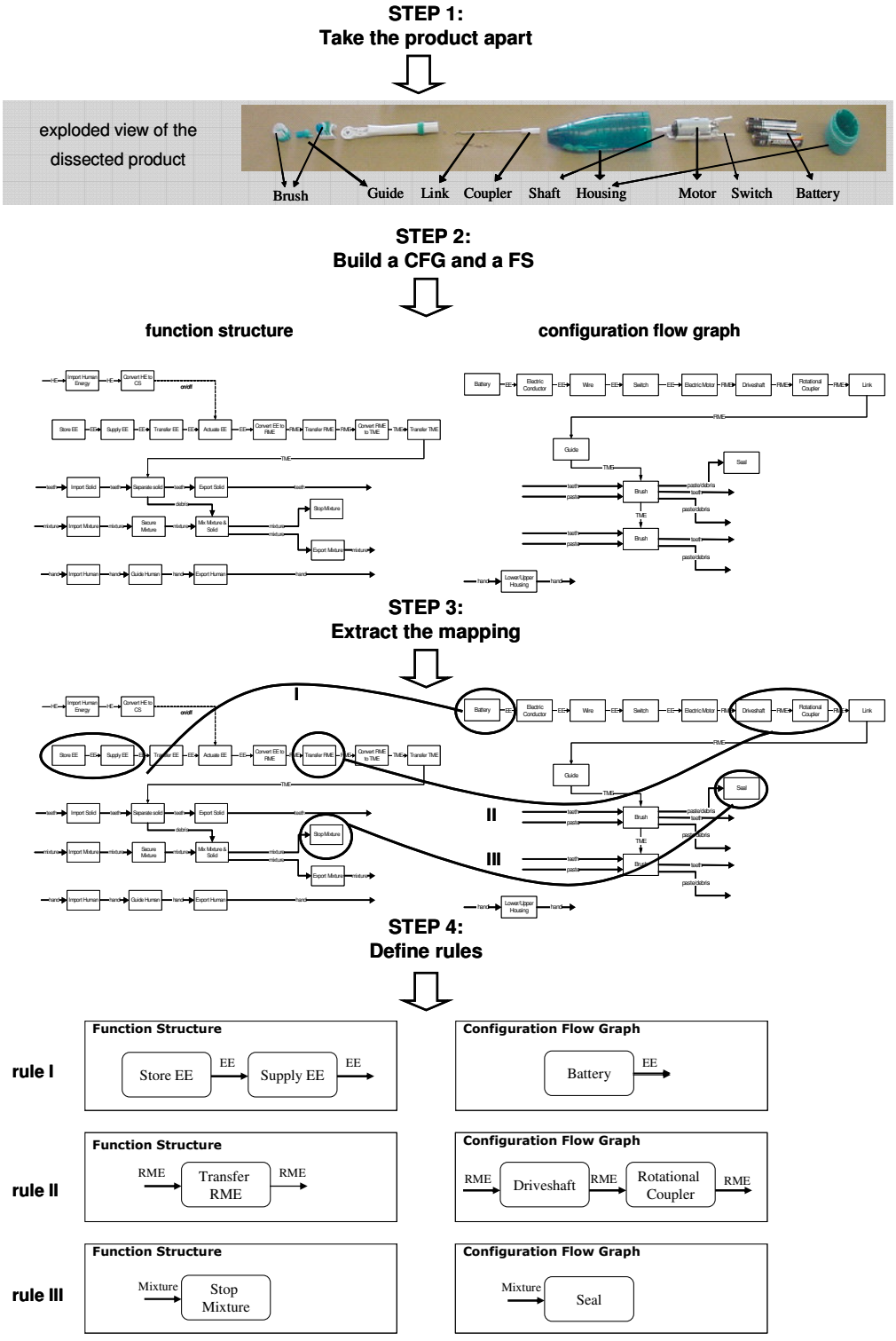


Figure 5.4 Illustration of the grammar rule derivation process

Note that the rules in the figure are not simply one-to-one matches of functions to components. The open-endedness of the grammar formulation allows escaping the tendency to assign single components to single functions. Instead, through multiple node recognition and application, the grammar provides a more generic approach capable of mapping multiple components for a single function, or a single component for multiple functions, as is the case in function sharing, or multiple components for multiple functions.

5.4 THE GRAMMAR RULE SET

In this section, the resulting grammar rule set is discussed. The main challenge in developing the rule set was to maintain the consistency of the rules derived. The aforementioned functional and component bases offer a significant remedy to attain this goal.

First, the level of granularity of the function structures has to be determined. In the functional basis, three levels of abstraction are developed to represent functions and flows: primary, secondary, and tertiary. To standardize function definitions in the grammar, the secondary level of the functional basis is adopted. Accordingly, functional modeling of the products is performed using function and flow terms shown in Tables 3.2 and 3.3 only. One exception to this is that of mechanical energy, which is represented at the tertiary level, allowing differentiating between rotational and translational mechanical energy.

Second, the configuration (component) knowledge captured by the rules is represented using the terms of the developed component basis. A thoroughly developed component taxonomy is critical to the implementation of the grammar, since the design decisions captured by the grammar rules are dependent on the component representation scheme followed. Both completeness and exclusivity criteria used during the

development of the component basis become important as they define the scope of the function-to-component mappings and the variety of the resulting solution space. Moreover, without a standardized component representation, the grammar rule set grows unboundedly, deeming the development of the grammar infeasible.

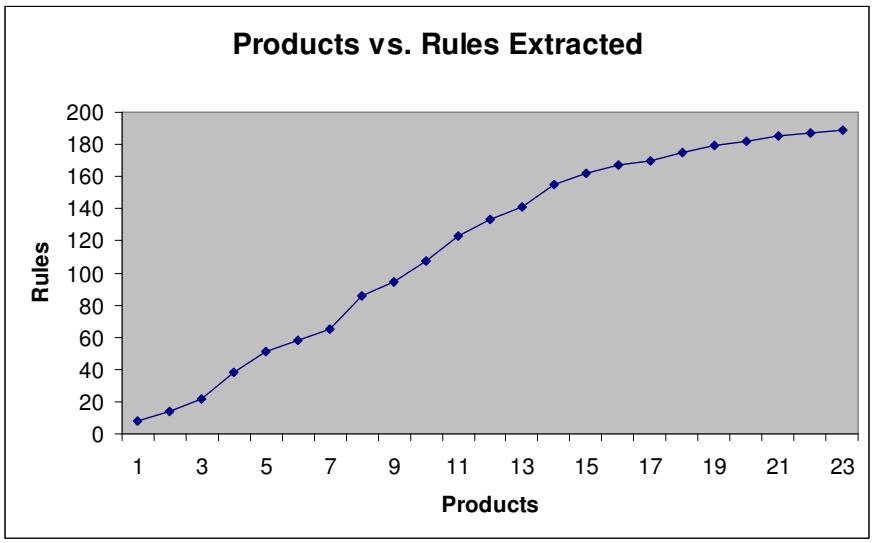


Figure 5.5 A plot of the number of products examined and the rules obtained from each of them.

Moreover, the construction of the rules is difficult since sometimes a rule may render another rule obsolete or invalid. Also, the same rule may be seen in successive products showing certain trends or standardized component solutions for certain functions. These cases are handled carefully such that only unique rules are added to the final rule set. Figure 5.5 shows the graph between products examined versus the rules obtained from them in chronologic order. Comparing the slopes of the right and left halves of the plot, one can see that the rate of newly defined rules is decreasing as more products are dissected. This observation suggests that a finite set of rules may describe the function-to-form mapping for the category of products chosen to be analyzed. The

current set of rules does not fully represent the entire set of engineering artifacts; however, it provides a framework for the method to be extended to other product categories or to other design contexts. Currently, there are 189 rules derived from 23 products. The complete list of 189 rules is shown in detail at http://www.me.utexas.edu/~adl/cfg_grammar.htm.

Chapter 6: Concept Generation Using the Grammar

The computational approach to design generation in this research is built upon the grammar formalism and is aimed at automating the function based design synthesis process. Accordingly, the functional description of a design is used as a starting point and the grammar permits automated generation of multiple conceptual solutions that address the functional requirements.

This chapter details the generation aspects of the developed computational design method. First, a brief overview of computational design synthesis methods is provided. This is followed by a detailed discussion of the automated concept generation process, and a summary of computer-based implementation of the grammar and the graphical user interface that is used.

6.1 REVIEW OF COMPUTATIONAL DESIGN SYNTHESIS METHODS

Researchers have employed different methods in order to computationally support the conceptual phase of design. In addition to the grammar-based approaches already reviewed, these methods include computer techniques such as constraint programming (Subramanian and Wang, 1995; Kota and Chiou, 1992), qualitative symbolic algebra (Williams, 1990), expert systems (Mittal, et al., 1985) or case-based reasoning (Navinchandra, et al., 1991; Qian and Gero, 1996; Bhatta et al., 1994).

Among these, one of the most historically significant is the expert system formulation described in the PRIDE system established by Mittal, et al. (1985) which is specifically developed for creating paper roller systems. A subset of expert systems, case-based reasoning techniques apply past knowledge stored in a computational database towards solving problems in similar contexts. It involves three stages: the representation of cases, the matching and retrieval of similar cases, and the modification of the reviewed

cases (Hsu and Woon, 1998). Examples include Gero et al. who presented a system called FBS (Qian and Gero, 1996) that uses relations among function, behavior, structure to retrieve design information to conduct analogy-based design. Similarly, the Structure-Behavior-Function modeling scheme (Bhatta et al., 1994) and its computational application KRITIK is a system relying on a design-case memory to conduct computational synthesis. Case-based reasoning techniques are suitable for problems where the retrieval and subsequent adaptation of solution cases are more computationally efficient than building a design from primitive elements. For those certain design domains where a large number of solution cases need to be stored this may not be practical. Moreover, case-based design systems are product knowledge oriented (as represented by case examples) and do not capture process knowledge in design.

Apart from expert system formulations, typical examples of computational synthesis applications start with a set of fundamental building blocks and some composition rules that govern the combination of these building blocks into complete design solutions.

Hundal (1990) designed a program for automated conceptual design that associates a database of solutions for each function in a function database. The user inputs the functions into a function structure in order to generate functional variants. The accuracy of a returned solution is dependent upon the designer's ability to break down the black box function of a particular problem to appropriate sub-functions and input/output quantities. The software is limited to generating solutions to functions contained in the available functional database.

Ward and Seering (1989) developed a mechanical design "compiler" to support catalog-based design. Built up from a database of "basic sets" of artifacts represented by a catalog number, the system takes appropriate schematics, specifications, and desired

functionality for a mechanical design as inputs and returns the catalog numbers for an optimal solution. The compiler eliminates catalog numbers that are incompatible with the previously implemented components.

Bracewell and Sharpe (1996) developed "Schemebuilder," a software tool using bond graph methodology to support the functional design of dynamic systems with different energy domains. The software utilizes a predefined functional embodiment knowledge base to seek solutions to the conceptual design functions. The system incorporates bond graph decomposition rules, a functional-embodiment database, and a component database and defines ports to characterize each component and limit connections to compatible energies. The designer enters the component function through FESTER (Functional Embodiment Structure–Extended Recursively), a function–means tree that supports hierarchical links that represent a process of function embodiment.

Chakrabarti and Bligh (1996) model the design problem as a set of input–output transformations. Structural solutions to each of the instantaneous transformation are found, and infeasible solutions are filtered according to a set of temporal reasoning rules. The temporal reasoning rules are only of a dynamic nature, i.e. they deal with force and torque issues, and are used to filter out the infeasible solutions.

Strawbridge, et al. (2002) developed a concept generation technique that utilizes a function-component matrix and a filter matrix to generate a morphological matrix of solutions for functions in a conceptual functional model. The function-component matrix (FCM) uses columns of components and rows of functions to characterize component functionality. Cells within the FCM matrix are either zero or non-zero depending on whether component j solves function i. An aggregate matrix can be constructed from individual product function-component matrices to generate a matrix describing the complete solution set. Following Strawbridge's work, Bryant, et al. (2005) developed an

automated concept generation tool that utilizes a repository of existing design knowledge. This work is similar in principle to the work presented here, but uses matrix-manipulation algorithms to generate and rank conceptual solutions.

Another function-based design synthesis approach is that of Xu et. al (2006). This research develops a design reuse framework that combines product modeling, knowledge extraction, and a computational search for concept generation and evaluation. In this research, the synthesis process is a three step transformation of customer requirements to conceptual design components. Using this approach, designs can be generated according to varying design objectives and constraints.

Another notable computational tool developed for concept generation is the agent-based system presented in the A-Design research (Campbell et al., 2000). The A-Design approach captures the interactions between individual components even if such interactions represent only partial configurations. It allows configurations to be created in either series or parallel. This research extended the representation developed by Welch and Dixon (1994) combining bond graphs (models of dynamic behavior; Paynter, 1961) with the design methodology posed by Pahl and Beitz. The representation developed here aims to relate components in terms of their dynamic behavior, shape, and purpose.

Several research efforts have focused on representing the interactions between artifacts within a product. Pimmler and Eppinger (1994) describe the interactions that occur between the elements of the product in terms of spatial, energy, information and materials. Interactions between components are ranked on a scale (-2,2), where -2 identifies that the interaction is detrimental for the desired product functionality and +2 identifies that the interaction is required for proper function of the product. The interactions between components are then used to define product architecture. However, the information modeled is not sufficient to be used at the concept generation stage.

Paredis et al. (2001) presents a method called "Composable Simulation" that utilizes ports to characterize each component. According to port characteristics, only compatible energy ports can be connected to each other. This framework is then used to analyze a system level design and to simulate the dynamic behavior.

Although useful, the methods described above for concept generation are not complete in the sense that they are either limited in their representation of function, form, or both. Bond-graph based methods are only limited to component types that deal with power transformation. Methods that are behaviorally oriented lack a general understanding of functionality in design limiting their applicability at the earliest stages. Moreover, none of the reviewed methods aim to model the design generation process and the decision making that is involved. The approach to computational design generation in this research follows the grammar formalism, and combines it with fundamentals of function based synthesis to model the design decision making that govern the creation of design solutions to functional requirements. Moreover, by integrating appropriate taxonomies (i.e. functional and component basis) to a graph grammar formulation, it offers a generic design language for capturing functionality, form, and their mapping in design. The details of this grammar-based generation approach are explained next.

6.2 GENERATION PROCESS AND THE RECOGNIZE-CHOOSE-APPLY CYCLE

The grammar provides an effective method for automatically generating design configurations through a search-based execution of rules. This computational synthesis approach is to perform a graph transformation of the initial function structure of the to-be-designed product into a set of configuration flow graphs. This process is illustrated in Figure 6.1. Each execution of a rule adds more components to the design configuration which incrementally builds to a final concept. At the end, the computational search

process returns different concepts with potentially varying degrees of complexity as *candidate* solutions to the same functional specifications.

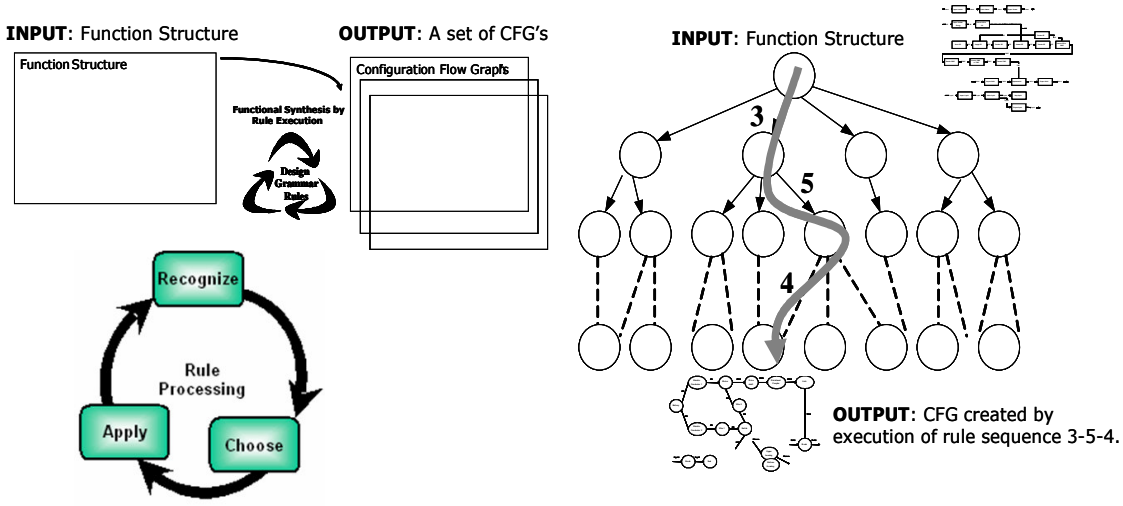


Figure 6.1 Illustration of rule processing via "recognize-choose-apply" cycle. A visualization of the search tree in building a CFG from a function structure

In detail, the transformation from the function structure to CFG is part of the recognize-choose-apply cycle shown in Figure 6.1. Given an initial function structure, S, first all possible rules that have their LHS as a subset of S are recognized. This step identifies all possible locations in the function structure where a grammar rule can be applied. These locations define a set of possible graph transformations that can be applied at that design stage. This recognition step is followed by a choosing of one of the valid options (i.e. graph transformation). In the final apply step, the CFG is updated as per the instructions provided in the RHS of the selected rule. This process is repeated until there are no more rules that can be applied.

The final candidate obtained at the end of this generation process depends on the selection of the rules applied. To fully automate the generation process, this selection is made by the computer. The basis and the guidelines to select the rules are embedded in

the search algorithm. In the current implementation, each applicable grammar rule is systematically selected by the computer with equal likelihood as the tree is traversed using a breadth-first search approach. An exhaustive search algorithm is selected, because finding a good conceptual solution requires an in-depth search of the design space and often necessitates generation of as many solution candidates as possible.

6.2.1 Designing a "Bread Slicer" Using the Grammar

In this section, the application of the grammar is illustrated for the design of a "bread slicer" device. The design process starts with the specification of the function structure of the product to be designed. This function structure for the bread slicer is shown at the top of Figure 6.2.

Note that this function structure is constructed using the secondary level of functional basis. This ensures consistency of the program input with the knowledge base of the design rules. If arbitrary function and flow naming is used in constructing the function structure, it has to be rephrased into the language of the functional basis before being inputted to the program.

The algorithm first reads the function structure and replaces sub-functions with components after application of each new rule until no further rules can be recognized. Figure 6.2 shows this process. At the beginning, none of the sub-functions are mapped to components. Therefore, the CFG starts as an empty graph. As rules are applied, the CFG is incrementally updated by the addition of new components. This is illustrated in the figure with four snapshots between start to finish. In between the snapshots, the grammar rules that are applied are listed. The highlighted sub-functions in the function structure designate the sub-functions that are mapped to a component solution. The result of this process is the completed design configuration shown at the end of the figure.

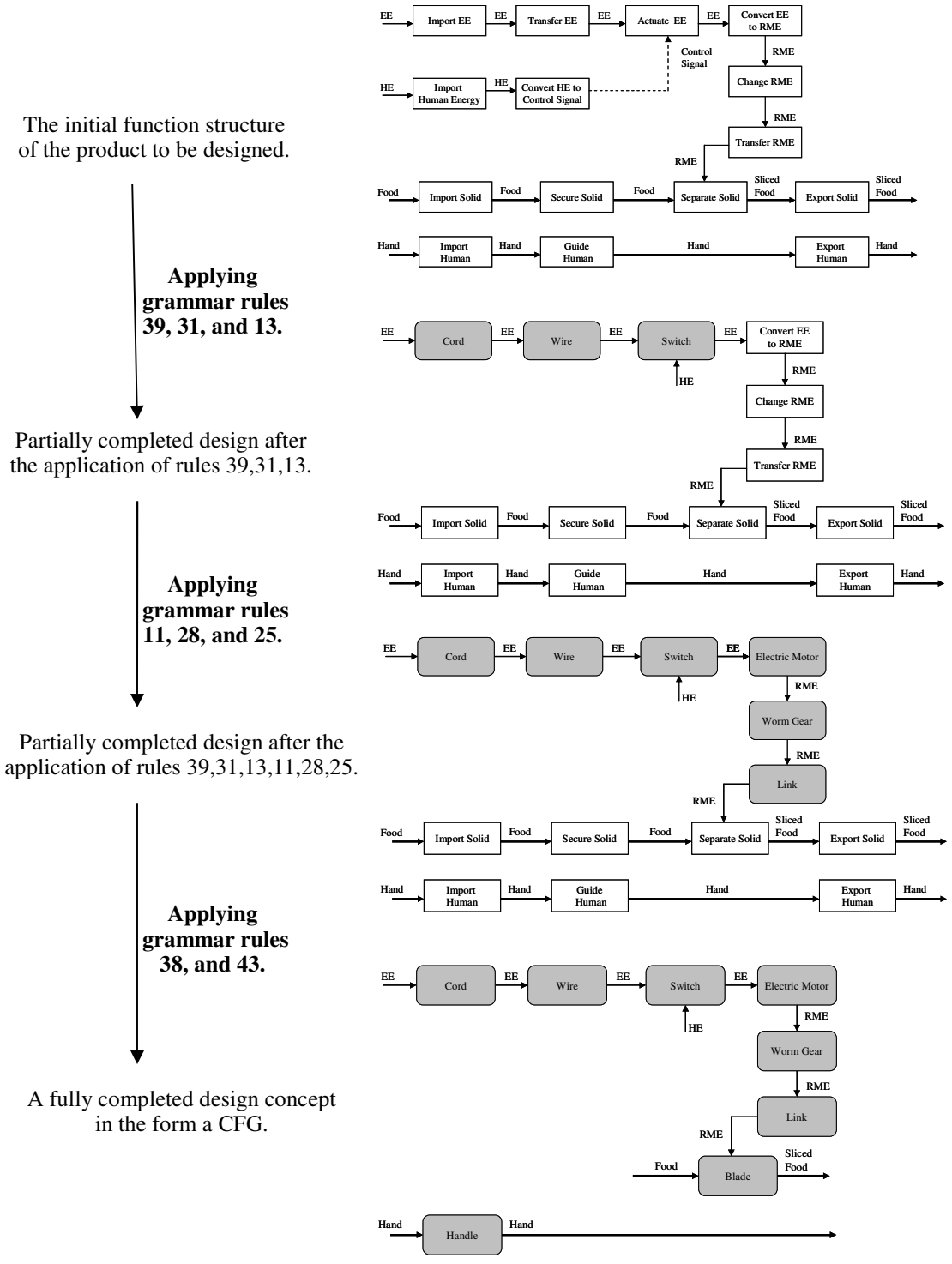


Figure 6.2 A pictorial representation of building a CFG from a function structure using the rule set.

In creating these design configurations, the program successfully integrates component concepts from different products into one complete concept variant. Consider the configuration shown at the bottom of Figure 6.2. In this conceptual solution, it is suggested that a gear-link pair be used for rotational energy transmission. This solution is leveraged from an original wine opener design, where the mechanical energy of a motor was transmitted to the cork through a worm gear and a rigid link. As a second example, consider the blade that is suggested to be used in the bread slicer device. This concept is borrowed from a paper shredder design, where a rotating blade is used to secure and shred the imported stack of papers. The configuration of Figure 6.2 also includes concepts inherited from an iced tea maker (cord), an electric iron (wire), an eyeglass cleaner (motor), a hand held vacuum cleaner (switch), and a dishwasher (handle). Examples such as these are very promising, because they show how the grammar approach can be extended such that a variety of concepts can be developed from a functional description of a product by synthesizing component solutions together that have been successfully used in the design of past products.

6.2.2 Ensuring Compatibility during Design Generation

Compatibility has to be ensured while synthesizing single or group of components of past designs into complete concept variants. To attain physical compatibility, the program employs a "flow" based compatibility measure. Specifically, a control strategy is enforced during CFG generation that only allows those sub-solutions to be connected to each other if and only if they share a common flow. For example, a pipe is not allowed to be connected to an electric resistor, because the output flow of one (hydraulic energy, or liquid material) is not the same type as the input flow of the other (electrical energy). Note, however, that these input-output flow types are not strictly prescribed for each component a priori. Instead, flow-based compatibility is enforced during graph

generation and is incorporated into the "recognize" step of the design generation cycle. This control strategy provides a required level of flexibility during design generation and facilitates the formation of novel connections that have not been seen in the past designs while maintaining a practical feasibility.

6.3 IMPLEMENTATION

The method described here is implemented in GraphSynth (Campbell, 2006). GraphSynth is a publicly available approach to create and implement graph grammars. Using GraphSynth, one can design, implement, test, and automatically invoke grammar rules that transform a graph from an initial seed into a creative design. Figure 6.3 shows the interface of GraphSynth.

Figure 6.3 Interface of GraphSynth (from www.me.utexas.edu/~adl/graphsynth/)

To perform the search method mentioned above, GraphSynth is integrated with a modified BFS search algorithm. This allows the designer to quickly draft a function structure that represents the conceptual design problem and create, and search for feasible conceptual configurations.

The program converts the initial function structure to XML format and starts the design generation process. Figure 6.4 shows the user interface with the initial function structure for a new design. The "input external" and "output external" nodes represent dummy nodes that ensure the input and output flows are not left dangling. Dangling arcs are not easily displayed in GraphSynth and are rarely considered as valid connections in many graph theories and applications.

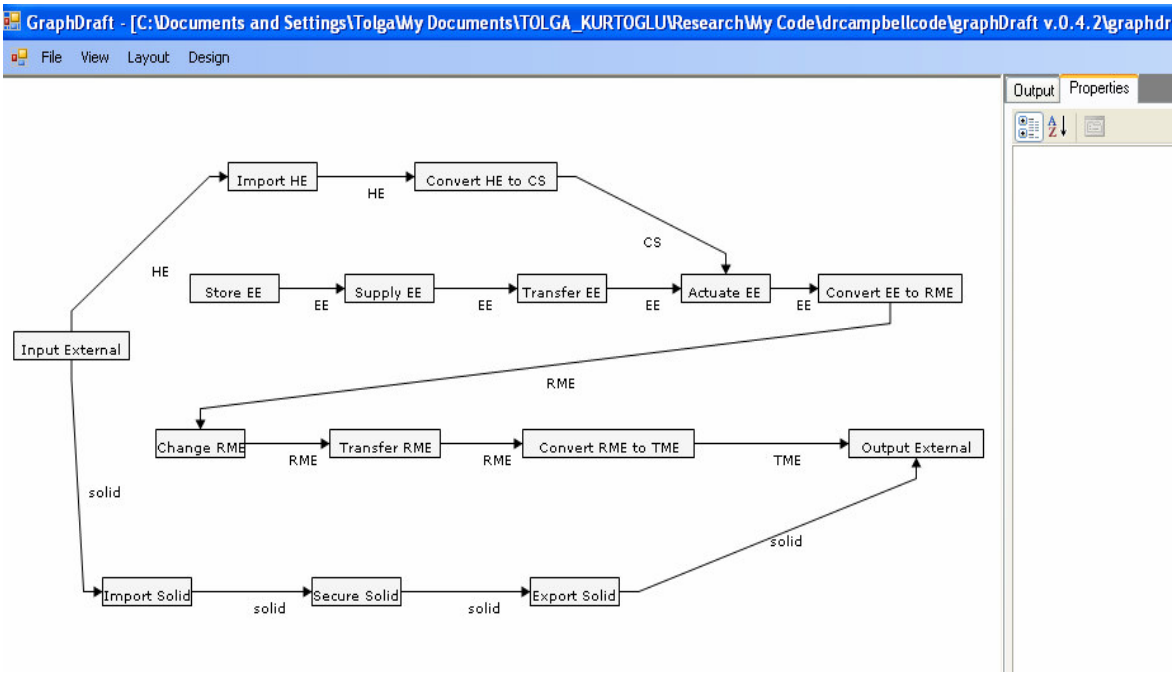


Figure 6.4 Graphical user interface of the automated concept generation software

The program then executes the recognize-choose-apply loop to implement the rules. At any instant, there are a number of rules that can be applied and the recognition

algorithm determines all of the possible rules as well as their locations with the function structure. This is accomplished by traversing the list of 189 rules and checking each for applicability. Any rule that satisfies recognition conditions is then listed as a "recognized" rule along with its corresponding location. In Figure 6.5, four grammar rules from the set of 189 are shown.

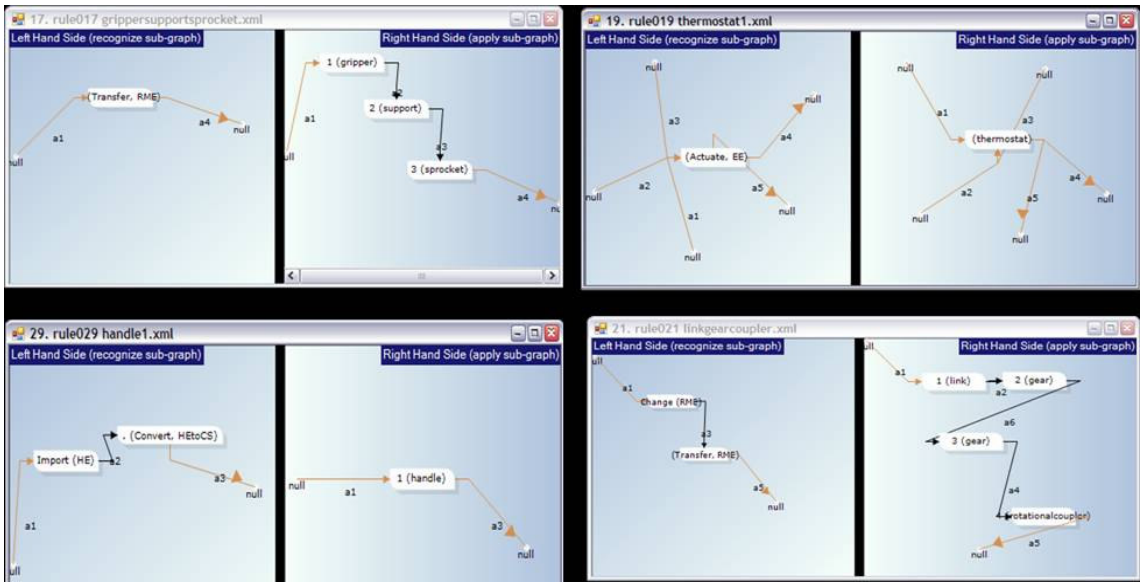


Figure 6.5 Four grammar rules of the conceptual design grammar

Recognition is followed by the selection of the rule to be applied. In the current implementation, the selection of different rule sequences (i.e. making different design decisions to create a variety of concepts) are explored through an exhaustive search that explores the design space for all possible configurations. Specifically, the computer employs a modified bread-first search (BFS) algorithm that includes a filtering mechanism which removes duplications of previously visited nodes from the search space. This algorithm takes advantage of a special property of the conceptual design grammar. Accordingly, the final candidate obtained is independent of the order of the rules chosen to be applied, a property known as "confluence" in graph grammar theory.

(i.e. execution of rule sequence 1-3-5 is the same as 1-5-3, or 3-1-5 providing that rule 1,3, and 5 are applied at same locations in all three cases.)

The result of this search process is a set of complete design configurations (CFG's) that span the entire solution space. This approach enables comprehensive exploration of design decisions (i.e. grammar rules) and removes any potential psychological bias toward certain design choices that may be present in traditional concept generation methods.

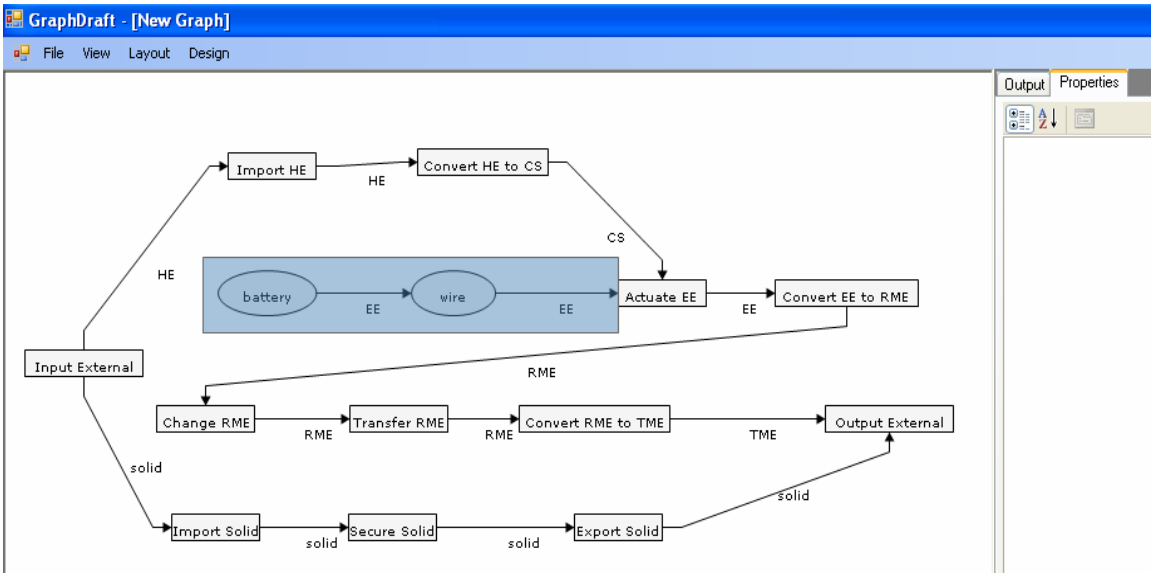


Figure 6.6 A screenshot of the user interface at a partially completed design state

The third and final step of rule processing is applying a selected rule. After the program makes a selection from the recognized rules list, it updates the function structure graph to a new configuration by replacing related sub-functions with their associated component(s) as described by the selected rule. This ensures that a sub-function is not recognized again, once it has been assigned a component solution. Figure 6.6 shows the state of the graph, after the application of "wire" and "battery" rules. In intermediate states, such as the one shown in Figure 6.6, the design is only partially completed.

Therefore, the graph in these states is a hybrid graph consisting of both function structure and configuration flow graph elements.

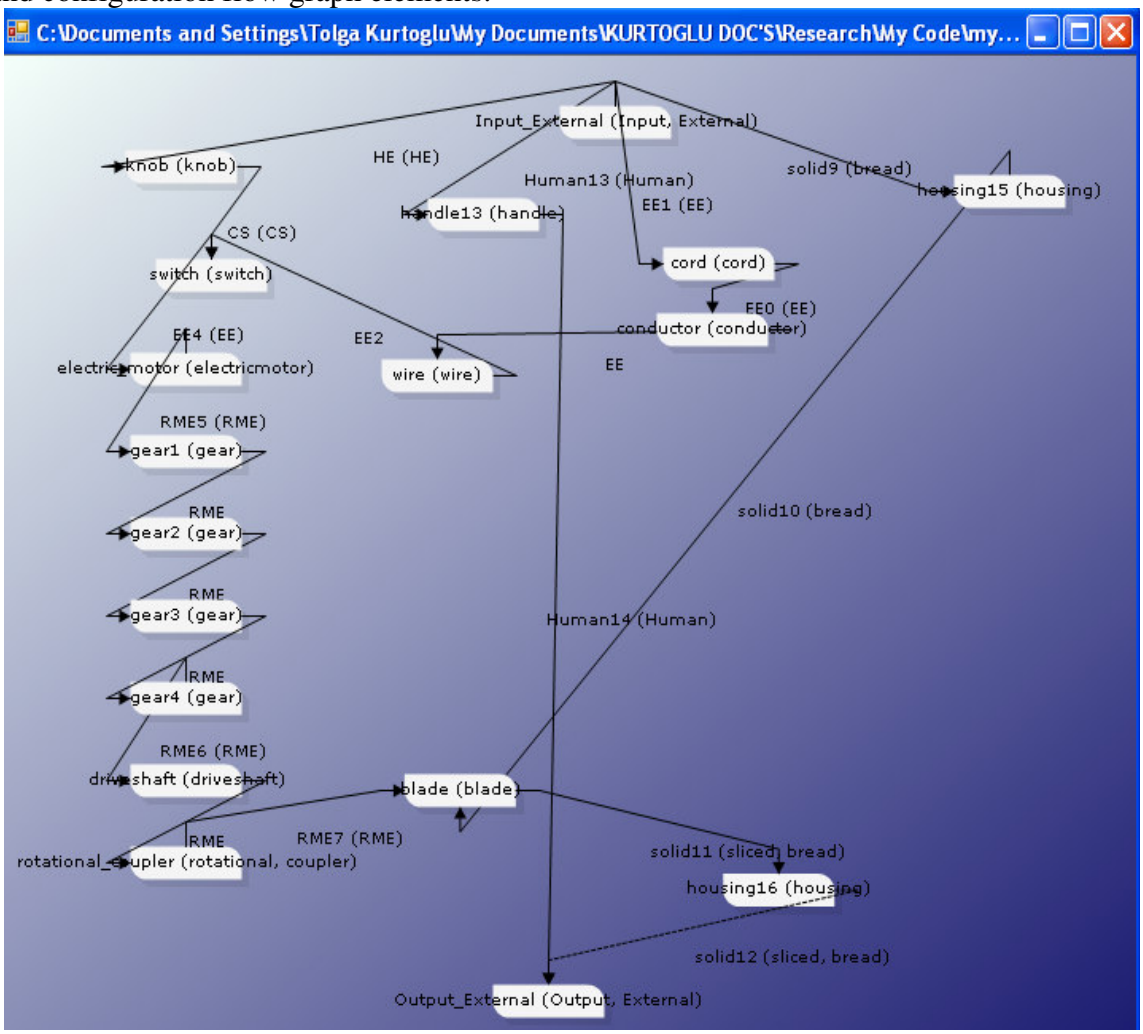


Figure 6.7 One of the 594 configuration flow graphs generated for the "bread slicer" design by the grammar

The program manages the rules and their applications until no further rules can be applied. Using the aforementioned breadth first search algorithm, the execution of different rule sequences leads to multiple conceptual solutions. Each of these candidates is unique and consists of a different selection and configuration of component concepts. For example, for the bread slicer design problem, the grammar generates 594 candidates

one of which is shown in Figure 6.7. The collection of these candidates describes the comprehensive space of feasible concept variants. The following challenge in conceptual design is the evaluation of these concepts. The method developed for design evaluation is presented next.

Chapter 7: Evaluation of Concepts Generated by the Grammar

As described in Chapter 1, design generation and evaluation are two tightly interconnected aspects of conceptual design. Finding a good solution usually requires an in-depth search of the design space, often necessitating generation of as many design alternatives as possible. These alternatives are then evaluated against various design requirements, constraints, and objectives to determine which alternatives are most useful for advancing towards successful designs and which alternatives have little or no design worth. Often, these two processes occur over many iterations until a satisfactory design solution is found. A closer look at the traditional and computer-based conceptual design methods reveals that computers and humans have distinct characteristics regarding design generation and evaluation (Simon, 1969). Humans are very good at comparing and evaluating solutions of various complexities by using sophisticated reasoning methods, however, it is impractical for them to generate all solution alternatives in a design space. In contrast, computers are well suited for generating numerous design alternatives, thanks to their computational speed, but they lack the judgment employed in human decision making to be able to effectively evaluate them.

In this chapter, a method is introduced that brings the designer and the computer together in order to leverage the strengths of both in generating and evaluating conceptual design alternatives. The objective of the developed evaluation scheme is to establish an interaction between a designer and the computational synthesis tool so that the designer's decision-making during concept evaluation can be analyzed, modeled, and later used for faster search of larger design spaces.

Accordingly, a tool is developed called DPM (Designer Preference Modeler) that facilitates communication between a designer and the previously described computational

synthesis software as shown in Figure 7.1. The computational synthesis tool generates design alternatives whereas the designer gets involved in the process by evaluating a prescribed set from these design alternatives. In order to get the synthesis software and the designer to interact, DPM carefully selects this set from the population of candidate designs and presents them to the designer for gathering evaluation feedback. This selection is made by following a heuristic that aims to simultaneously reduce the number of required designer evaluations and capture the variety in the design solution space. The designer's feedback is translated into a preference model that is used to automatically search for best designs. Because the preference model is built directly from the evaluations of the designer, it reflects the designer's reasoning and judgment under specific design objectives and constraints. The main advantage of the method is that it allows the construction of a designer preference model from limited number of designer evaluations, which then can be used for evaluating large number of design alternatives.

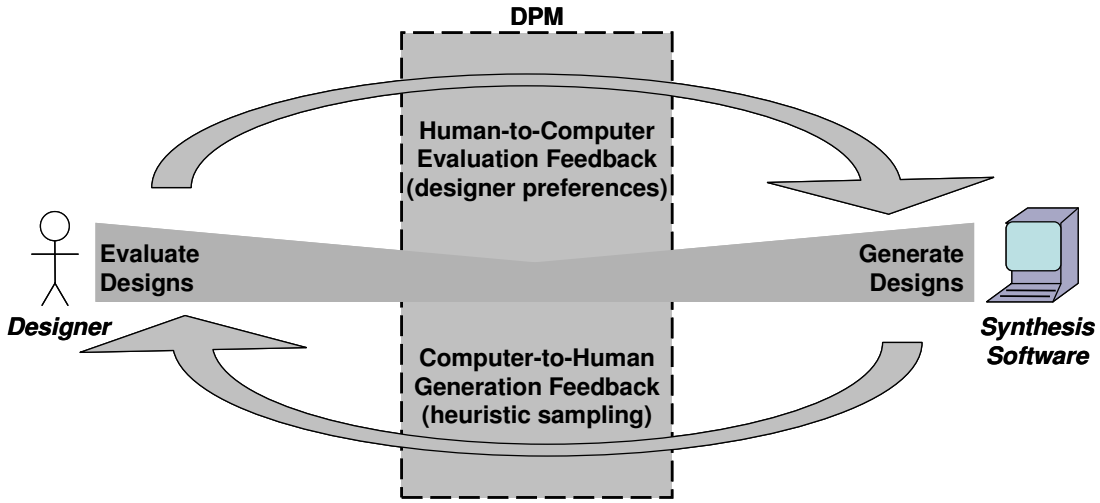


Figure 7.1 DPM brings the designer and grammar based computational synthesis tool together so that the designer's decision making during concept evaluation can be modeled

The rest of this chapter presents the state of the art in design evaluation, and learning based design techniques. This is followed by a description of the method used by the DPM and a presentation of its implementation through an illustrative example.

7.1 REVIEW OF EVALUATION AND SELECTION METHODS IN DESIGN

Design evaluation and selection is an important part of the design process and has received great attention in the design literature. In this section, a review of various decision-based design techniques (Hazelrigg, 1996) is provided and traditional and computer-based methods for evaluating a pool of conceptual solution alternatives are described.

Perhaps, the most common concept evaluation method is the Pugh Concept Selection Charts (Pugh, 1991). Pugh charts use a minimal, qualitative evaluation scale and compare design alternatives in a matrix format against a number of performance criteria. Pugh Charts provide an effective known tool for preliminary concept selection when there is minimal information available about the potential design solutions. Numerical concept scoring/weighting, and decision matrices (Ullman, 1995) are similar methods and only vary in the representation of the nature (qualitative vs. quantitative) and the resolution of their evaluation scales.

The Analytic Hierarchy Process (AHP) (Saaty, 1980) is another multi-criteria decision making technique that uses hierarchically related performance metrics. Similar in spirit to AHP, some researchers have proposed methods that are inspired from the multi-attribute utility theory (Keeney and Raiffa, 1976). The general method for these techniques is to assign a value for each performance metric, weight the value by the importance of the metric, and then aggregate the weighted scores to convert multiple metrics into a single metric. Application of these methods to design evaluation include formulation of non-linear utility functions capturing multi-attribute aspects of

engineering problems (Thurston, 1991), physical programming (Messac, 1996), and set-based techniques that rely on fuzzy logic to represent imprecision in design and to conduct design evaluation (Wood et al., 1990).

Finally, at later stages of design where simulation data is available, computational analysis tools such as finite-element-methods (FEM), computational fluid dynamics (CFD), etc. offer accurate and robust evaluation of design alternatives. The integration of these different analysis tools into a single, robust evaluator which can negotiate multiple attributes of a design problem remains a challenging topic and is tackled by the field of multi-disciplinary design optimization.

7.2 REVIEW OF LEARNING-BASED METHODS IN DESIGN

Examples of learning-based computational design tools include the Learn-It (Stahovich, 2000) and its temporal extension Learn-It-II (Rawson and Stahovich, 2006) systems which observe a designer's actions and use an instance-based technique to learn the design strategy employed. Both these systems are intended for parametric design problems in which the designer iteratively adjusts the parameters of a design to meet specific design requirements. The learned strategies are later used to automatically generate design solutions when the design requirements change.

Myers et al. (1999) have created a system that monitors a designer's interactions with a CAD tool in order to automatically produce design documentation. This system however, does not perform automated design. Moss et al. (2004) integrated a learning mechanism to the agent-based computational design tool called A-Design (Campbell et al., 2000) in order to enable the system to learn from its own design generation experiences. However, A-Design does not get feedback from the designer in the process and its learning scheme is aimed at improving the quality of designs generated by its agents.

7.3 REPRESENTATION OF CANDIDATES AND THE RECIPE OF A CONFIGURATION

As described in Chapter 6, the result of the computational generation process is a set of design configurations. Of this set, each design configuration is represented by a list called the *recipe* that captures the sequence of rules that are used in the construction of that particular configuration. For example, the recipe for the CFG shown at the bottom of Figure 6.2 is {39, 31, 13, 11, 28, 25, 38, 43}. Depending on the grammar rules chosen to create the CFG, the recipes of different solutions may have potentially different lengths to their lists. The use of the recipes by the DPM is explained in the next section.

7.4 CONCEPT EVALUATION USING THE DESIGNER PREFERENCE MODELER

Design evaluation provides a means of determining the 'value', 'usefulness', or 'strength' of a design solution with respect to a given objective (Pahl and Beitz, 1988). An evaluation requires a comparison of concept variants, or a rating of solutions based on an idealized design solution. DPM facilitates the evaluation of design concepts using an interactive approach that involves the designer in this critical process. Specifically, the designer is involved in two ways. First, the designer is given flexibility to define his/her own evaluation criterion. Secondly, the designer's preferences based on the criterion are captured as preference scores of the rules that are used during the creation of designs. In other words, DPM models how much the designer prefers a particular rule over others in the grammar, or synonymously how much the designer likes a particular function-to-component mapping as described by that rule.

DPM carries out the construction of this model by employing a sampling strategy that selects and presents the designer with a small-set of proposed solutions. It performs this sampling by following an algorithm that aims to simultaneously reduce the number of required designer evaluations and capture the variety in the design solution space. The goal of the sampling is to find the minimum set of solutions, recipes of which represent

the most diverse collection of the rules (i.e. solution principles, or components). After completing the sampling, DPM presents the selected solutions to the designer for evaluation. The designer is asked to make pairwise comparisons between the selected designs. These solutions are then rated by the designer based on specific design objectives and constraints. After the solutions are scored, the solution ratings are projected on to the rules. Using this projection, a "preference score" is assigned to each rule. The collection of these preference scores constructs a model of designer preferences. Finally, DPM uses the model of designer preferences to guide the search for finding the "best" configurations in the population of generated solutions. To achieve that, preference scores of rules are propagated over each recipe to compute the overall worth of each design in the population.

Some definitions used in the formulation of the evaluation method are as follows:

c_i Є CP is a candidate;

An automatically generated configuration (CFG) as a solution to the design problem.

CP = (c_1, c_2, c_3, …, c_n) is the candidate pool;

The collection of all candidates where n is the number of candidates generated.

DFS = (c_i, c_j, ….,c_k) is the designer feedback set; where DFS is a subset of CP

The set of candidates selected to be presented to the designer for evaluation feedback.

r_j is a rule;

A grammar rule (function-to-form mapping) used in the generation of a candidate.

R_i is the recipe of the candidate c_i;

The collection of rules used in the generation of candidate c_i.

$RP = R_1 \cap R_2 \cap R_3 \cap R_4 \cap R_n$ is the recipe pool;

The union of recipes of all candidates where n is the number of candidates generated.

$CDFS = (s_i, s_j, \ldots, s_k)$ is the cumulative designer feedback score vector;

The vector of cumulative scores of candidates in DFS after designer evaluations.

$RPS = (s_{ri}, s_{rj}, \ldots, s_{rk})$ is the rule preference score vector;

The vector of preference scores of rules in RP.

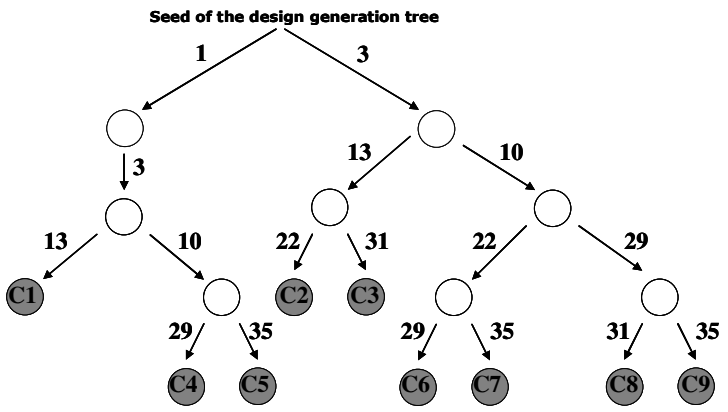
$CS = (cs_i, cs_j, \ldots, cs_k)$ is the candidate score vector;

The vector of candidate scores of candidates in CP.

7.4.1 Sampling Strategy

The goal of the sampling is to select a set of candidates from the candidate pool. This set called the designer feedback set (DFS) will be presented to the designer for obtaining evaluation feedback. As one might expect, there may be countless possibilities for choosing the DFS. Here, DPM employs a heuristic sampling strategy that strives to strike a balance between two objectives. First, it aims to choose candidates that will keep the number of designer evaluations to a minimum. It accomplishes this by taking advantage of the commonality of candidates in the candidate pool. Accordingly, DPM selects candidates with a high "commonality measure". Simultaneously, it targets candidates that will best represent the variety in the solution space. To achieve the latter,

DPM keeps track of a "variety measure" for the rules that constitute the recipes of the candidates in the DFS and only allows addition of the candidates that increase the variety of rules in the DFS. By monitoring these two measures, DPM continues to select candidates from the candidate pool until all rules in the recipe pool are represented in the DFS.



(a)

Candidate	Recipe
C1	(1,3,13)
C2	(3,13,22)
C3	(3,13,31)
C4	(1,3,10,29)
C5	(1,3,10,35)
C6	(3,10,22,29)
C7	(3,10,22,35)
C8	(3,10,29,31)
C9	(3,10,31,35)
Recipe Pool	(1,3,10,13,22,29,31,35)

(b)

Figure 7.2 Example of the application of the sampling strategy

To better understand the sampling approach, consider the example shown in Figure 7.2-7.5. This is a design example illustrating the synthesis of a switch assembly in a soda maker. The design tree shows the generation of the candidates. The arc labels in the tree correspond to the grammar rule that is applied during the transition between any two nodes of the tree. The basic steps in applying the heuristic sampling strategy are as follows:

1. Compute the candidate pool (CP), and the recipe pool (RP).

As shown in Figure 7.2a, the final design configurations, i.e. candidates, are represented at the leaves of the tree and numbered as C1-C9. The recipes of the nine candidates and the recipe pool are summarized in Figure 7.2b.

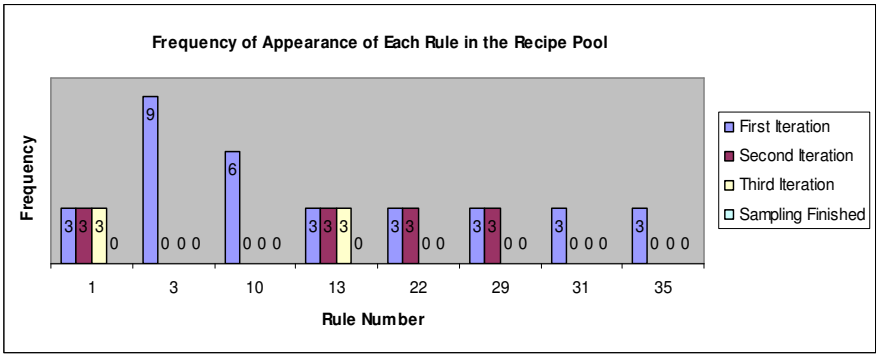


Figure 7.3 Frequency of appearance of rules at each iteration

2. Create a histogram of the frequency of rules in RP.

Figure 7.3 shows the "frequency of appearance" of the rules at each iteration. Rule#3 is used in the construction of all nine designs and has the highest frequency with 9 in the first iteration.

3. Calculate the commonality measure of each candidate by summing frequencies of rules in the recipe of the candidate.

Figure 7.4 shows the "commonality measure" of the candidates at each iteration. In the first iteration, C4-C9 has the highest commonality measure with 21.

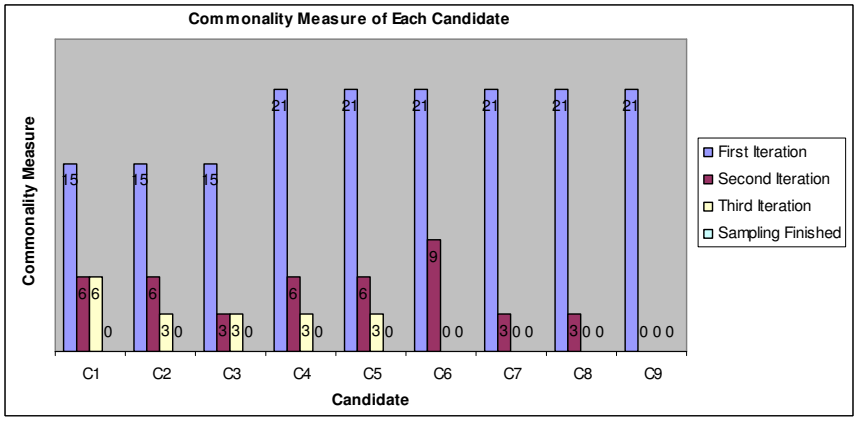


Figure 7.4 Commonality measure of rules at each iteration

4. Select the candidate with the highest commonality measure and add it to the DFS.

Among the four candidates with the highest commonality measure, C9 is (randomly) selected and added to the DFS.

5. Remove the rules of the selected recipe from the recipe pool.

After the selection of C9 to the DFS, the rules in the recipe of this candidate (3, 10, 31, 35) are removed from the recipe pool. As can be seen in Figure 4.c., the frequencies of the rules 3, 10, 31, and 35 following the first iteration are zero. This ensures that there will be minimum overlap between C9's recipe and the recipes of future selections. This modification is a necessary step for minimizing the number of candidates included in the DFS at the end of the sampling.

6. Repeat steps 1-5 until no rules remain in the recipe pool.

Following a similar scheme, C6 is selected in the second (with a commonality measure of 6), and C1 is selected in the third (with a commonality measure of 6) iteration. The result of the sampling process is the DFS containing C9, C6, and C1. This DFS satisfies the two objectives of the heuristic sampling of finding a set with minimum number of candidates, recipes of which represents the most diverse collection of the rules. Accordingly, the generated DFS captures all the rules in the recipe pool by the selection of only three candidates.

7.4.2 Designer Feedback

The sampling is followed by gathering the designer's feedback. To achieve this, DPM presents the candidates in the DFS to the designer for evaluation. The designer is asked to make pairwise comparisons between the selected candidates. The evaluation scale utilized during this concept scoring is similar to the one used in decision matrices (Ullman, 1995), and contains the ratings {-3, -2, -1, 0, 1, 2, 3}. When element i compared to j is assigned one of the aforementioned ratings, then element j compared to i is

assigned its negative. The GUI for this step is shown in Figure 5. The overall worth of each candidate as a result of this evaluation, called the cumulative designer feedback score (CDFS), is calculated by simply adding the values in the cells of the column that corresponds to that particular candidate.

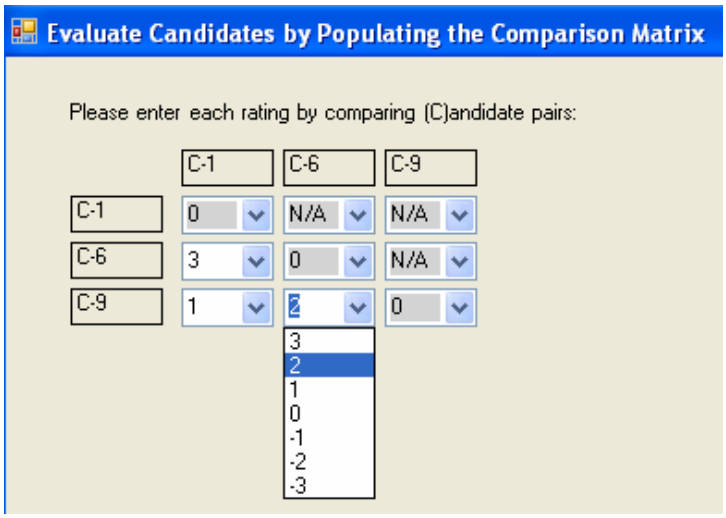


Figure 7.5 The GUI of the pairwise comparison matrix used during designer evaluations.

Considering the values in Figure 7.5, the cumulative designer feedback score for candidate C1 becomes 4 (0+3+1). For candidate C6 this score is -1 (-3+0+2), whereas for candidate C9 it is -3 (-1-2+0). These scores form the designer feedback score vector [4, -1, -3].

7.4.3 Construction of the Designer Preference Model

After the designer completes the pairwise comparisons between the candidates, the candidate ratings are projected onto the rules in the recipe pool to compute a "preference score" for each rule. The collection of these preference scores constructs a model of designer preferences as represented by the rule preference score (RPS) vector. The basic steps in the computation of the RPS vector are as follows:

1. Calculate the cumulative designer feedback score for a candidate in the DFS by summing each column in the comparison matrix,

2. For each rule in the recipe of a candidate, add the cumulative designer feedback score of that candidate to the rule score,

3. Repeat step 1-2 for each candidate to calculate the final rule preference score s_{ri}

4. Aggregate the rule preference scores to construct the RPS vector.

Considering the same example shown in Figure 7.2-7.4 and taking the cumulative designer feedback scores calculated in the previous section, the value of 4 is added to the rule score of the rules in the recipe of C1 (rules 1,3, and 13). Similarly, the value of -1 is added to rule score of the rules 3, 10, 22, and 29 and the value of -3 is added to the rule score of the rules 3, 10, 31, and 35. Aggregating these values defines the RPS vector, which then becomes: RPS = [4, 0, -4, 4, -1, -1, -3, -3] corresponding to the "preference scores" for rules 1, 3, 10, 13, 22, 29, 31, 35. The RPS vector represents a model of the designer's preference for each rule in the recipe pool. In this particular example, the designer prefers rule#1 and rule#13 the most, and rule#10 the least.

7.4.4 Automated Concept Scoring Using the DPM

DPM uses the model of the designer preferences to automatically search for the best designs in the population of generated candidates. This is performed by propagating the preference scores of rules over each candidate recipe in the population. Specifically, by summing over the individual rule preference scores in a recipe, the worth of each candidate can be computed. Finally, the aggregation of these scores constitutes the candidate score (CS) vector. After all candidates are evaluated, they can be ranked or sorted for identifying the best or the worst designs in the population.

Returning to the switch assembly design, and using the RPS vector shown before, all of the nine candidates in the population can be evaluated. The CS vector for this example then becomes: CS = [8, 3, 1, -1, -3, -6, -8, -8, -10] corresponding to the "candidate scores" of candidates C1-C9. The best three designs in this population are C1, C2, and C3. Note that in evaluating the candidates, DPM preserves the designer's original ranking among the candidates in the DFS, i.e. the relation C1 > C6 > C9 holds true after the DPM evaluates the candidates using the designer preference model.

7.4.5 Evaluating "Bread Slicer" Designs Using the DPM

In this section, the evaluation scheme of the DPM is demonstrated using the "bread slicer" design problem introduced in Chapter 6. Recall that the function structure for this design (shown at the top of Figure 6.2) includes 15 sub-functions, and 19 flows and the grammar-based generation process creates 594 valid design configurations as proposed solutions to the given functional specifications. The goal of the evaluation scheme is to find the *best* design in this population of 594 designs.

Following the procedural steps of Section 7.4.1, a candidate pool is generated by the DPM. After the candidate pool is generated, DPM employs its sampling algorithm and selects 12 candidates from the candidate pool. These candidates are added to the DFS and are presented to the designer for evaluation feedback. The designer evaluates the 12 candidates based on the criteria of quality of concepts. Finally, the rule preference score (RPS) and the candidate score (CS) vectors are computed by the DPM. The result of this process is summarized in Figure 7.7, and the best and the worst candidates determined by the evaluation are shown in Figures 7.6 and 7.8.

Looking at these two candidates, one can see that the design of Figure 7.6 has a number of superior features when compared to the design of Figure 7.7. First, it includes a knob component to activate the switch which can be used for setting the speed of the

slicer. Second, the power transmission from the electric motor to the blade is much better developed in this design. It includes a gear box for speed reduction, and a rotational coupler that secures a blade to the output shaft of the gear box. Moreover, in this design, the blade is supported by an outer housing that helps with importing and securing of the bread into the device. This was a feature that the designer particularly favored while providing evaluation feedback for reasons regarding the customer needs of "safety" and "ability to accommodate different slice sizes". On the contrary, in the other design, the importing and securing of the bread are addressed by the blade itself, similar to how a traditional electric kitchen knife functions.

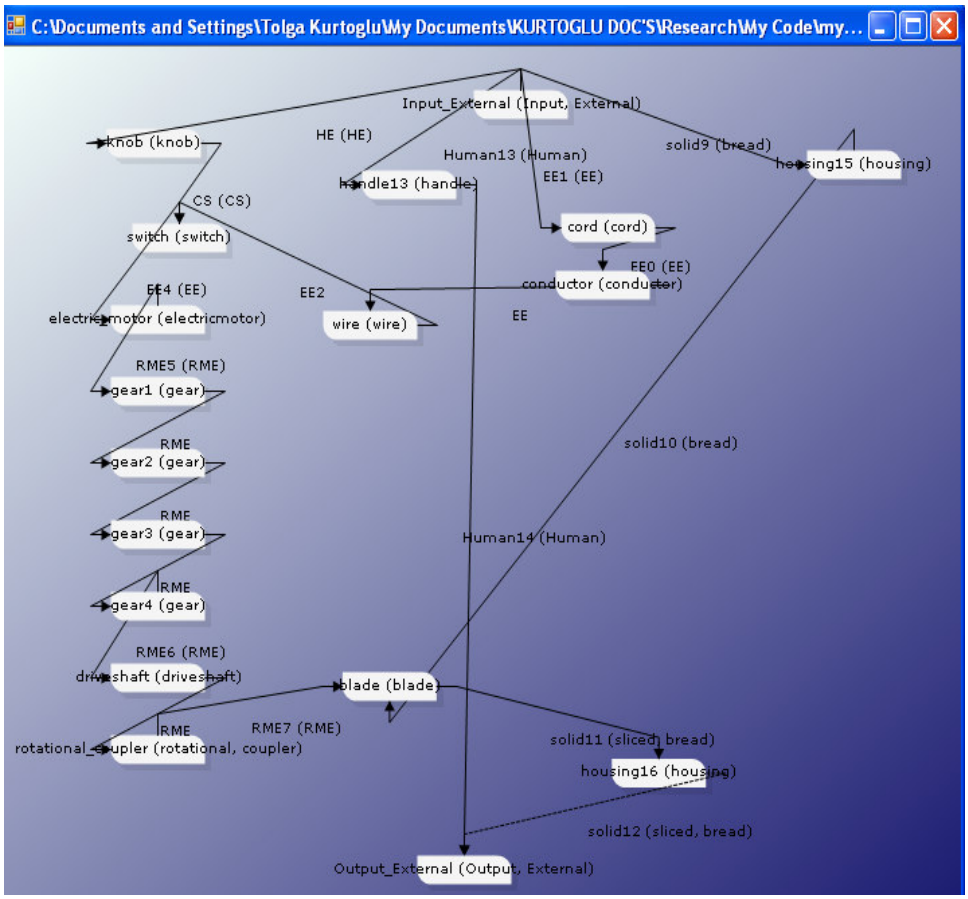


Figure 7.6 The best candidate in the population for the "bread slicer" design

Design Problem			
Example Problem	*Bread Slicer*		
Evaluation Criteria	Quality of Concepts		
Design Generation and Sampling			
# of candidates generated	**594**		
# of rules in the recipe pool	30		
# of candidates in DFS	12		
Design Evaluation			
Recipe Pool	1,7,8,10,11,12,13,15,17,20,21,22,23,24,25,26, 28,29,30,31,33,34,35,36,38,39,40,41,42,43		
RPS	-3,3,14,3,0,6,-3,7,-21,-7,15,3,24,-9,-14,7, -3,3,-9,0,2,-9,0,3,0,0,9,9,9,-9	***Recipe***	***Score***
Best Candidate	C155	10,11,15,22,23,29,38,39,40,41,42	**67**
Worst Candidate	C498	1,8,11,13,17,38,39,43	**-50**

Figure 7.7 The tabulated results for the "bread slicer" design problem

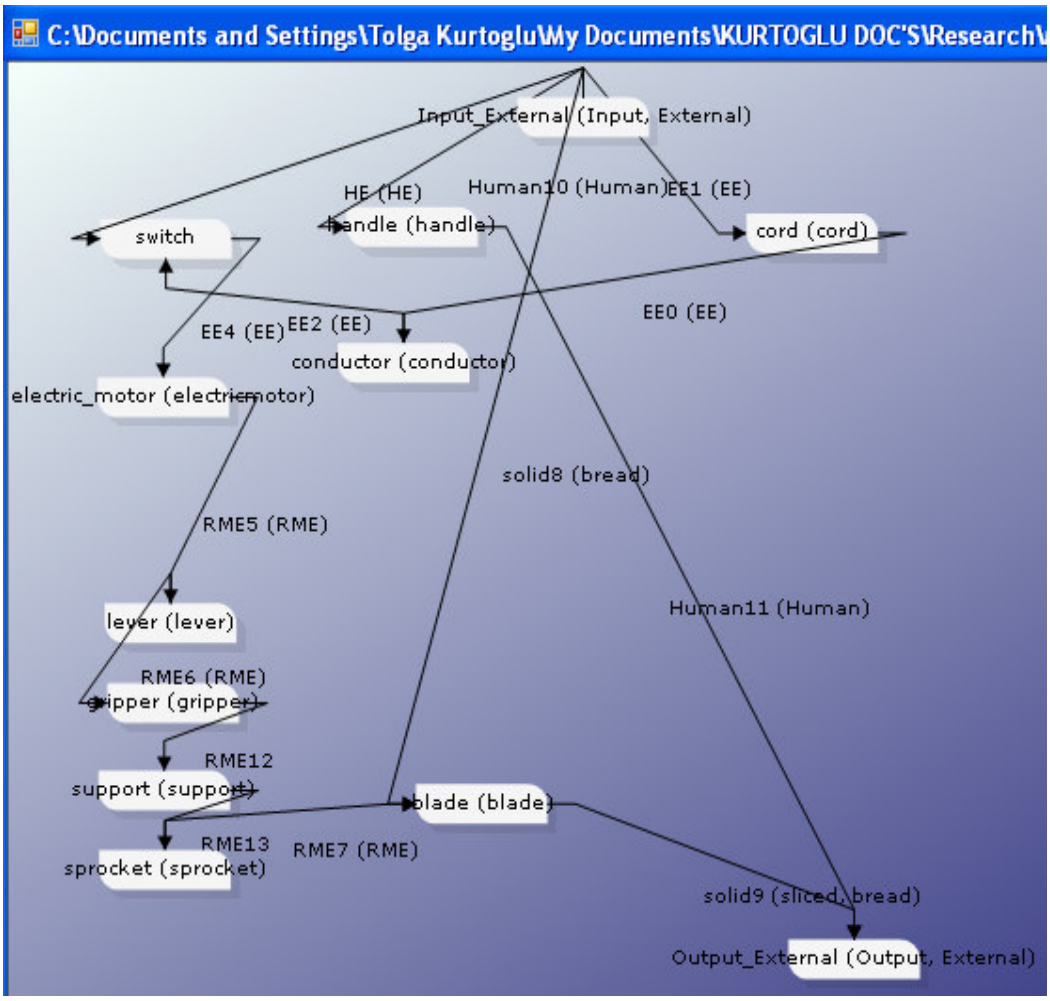


Figure 7.8 The worst candidate in the population for the "bread slicer" design

As this case illustrates, DPM generalizes a small number of designer evaluations into a preference model, which it later uses to evaluate a large range of automatically generated design alternatives. In the next chapter, additional electromechanical design cases are presented to illustrate the capabilities of the conceptual design grammar and the design preference modeler.

Chapter 8: Electromechanical Test Cases

This chapter presents the application of the computational method on two electromechanical test problems. These design problems are selected to mimic industry level design challenges. The implementation includes both the concept generation and the concept evaluation phases. The results are provided at the end along with a discussion of the implications of these two examples.

8.1 DESIGN PROBLEM I: BOTTLE CAPPING MACHINE

The first problem is to design a device that registers a bottle to a capping station, caps the bottle, and allows somebody to retrieve the capped bottle from the device. In order to pose the problem, the user supplies a conceptualized function structure of the to-be-designed device. Figure 8.1 shows this function structure.

This function structure should be constructed by the designer after studying the problem description and particular design requirements. Providing the function structure reflects certain decisions of the designer and imposes initial constraints on the design solution. For example, for the function structure of the bottle capping machine shown, a decision has been made to “secure the bottle” and “translate the cap” before the capping operation. This relative motion scheme is a preference of the designer and could be reversed if one decides to “translate the bottle” to the capping station and to “secure the cap” before the capping action. The particular selection made is due to the relatively lighter weight of the cap compared to the bottle. Thus, it has been conceptualized that translating the cap would require less power – an objective of the design problem. Similar decisions have been made for other aspects of the design during the construction of the function structure shown in Figure 8.1.

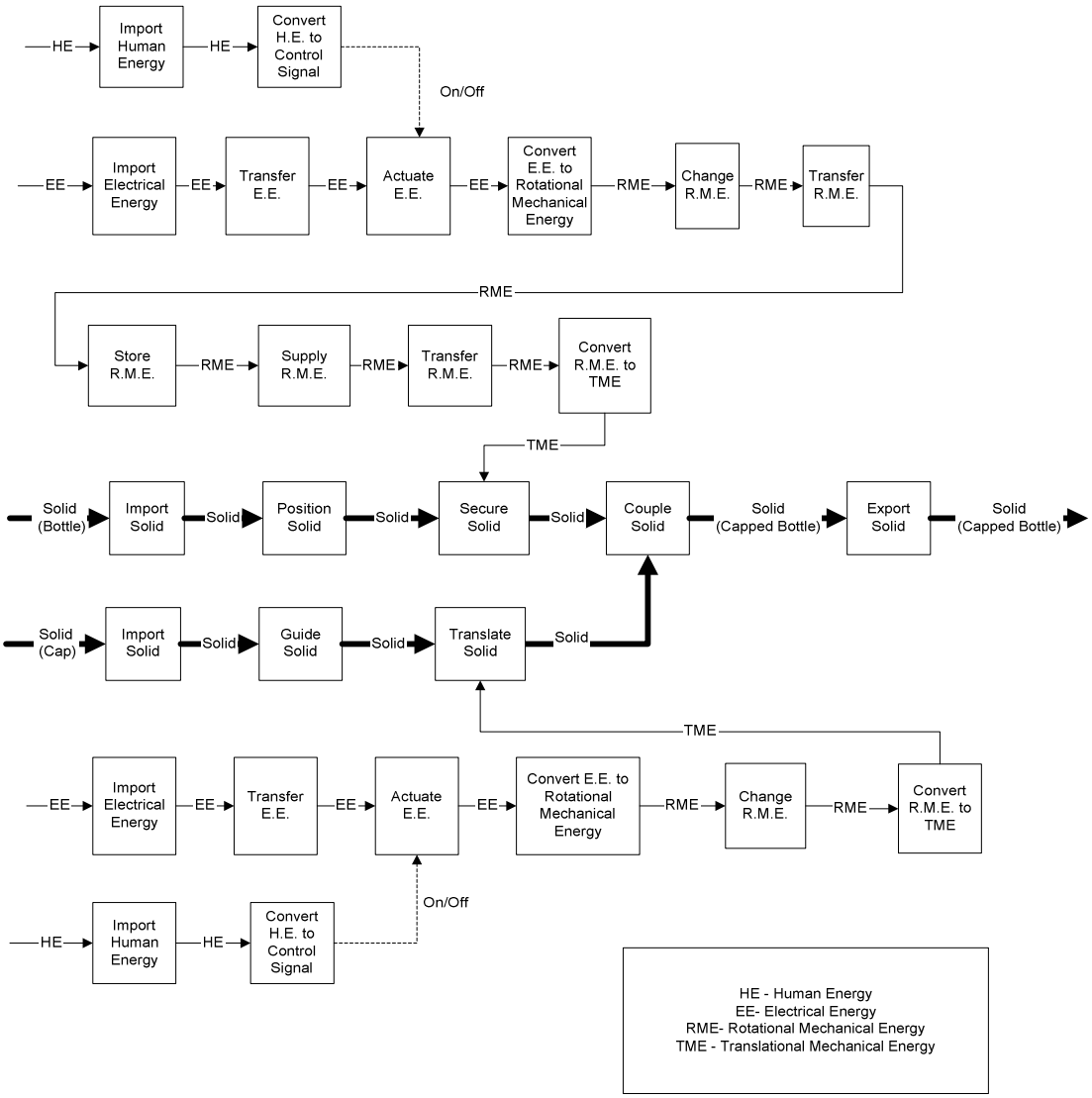


Figure 8.1 The function structure for the bottle capping machine

Since the generation process is already been described in the previous chapters, the results are presented next. The grammar generates 339,168 unique design configurations for function structure (28 sub-functions, and 34 flows). The generation results are tabulated in Figure 8.2. Of the 189 rules of the grammar rule base, 42 are used in the generation of these configurations. These 42 rules are recognized at a total of 53 different locations in the function structure graph. This is a result of some rules being

recognized in more than one location in the host graph. The resulting candidate set includes configurations consisting of 28 different component concepts. These components are listed in Figure 8.2. Two designs generated by the grammar are shown in Figures 8.3 and 8.4.

Design Problem		***List of Components***
Example Problem	*Bottle Capping Machine*	conductor, switch, wire, cord, electric motor, knob, handle, belt, gear, shaft, agitator, rotational coupler, sprocket, support, lever, link, clamp, worm gear, pulley, spring, housing, key, punch, container, sled, guide, piston, hinge
# of Subfunctions in Input	28	
# of Flows in Input	34	
Design Generation Details		
# of different grammar rules invoked	42	
total # of grammar rules invoked	53	
# of candidates generated	**339,168**	
# of different components used in synthesis	28	

Figure 8.2 The generation results for the bottle capping machine design

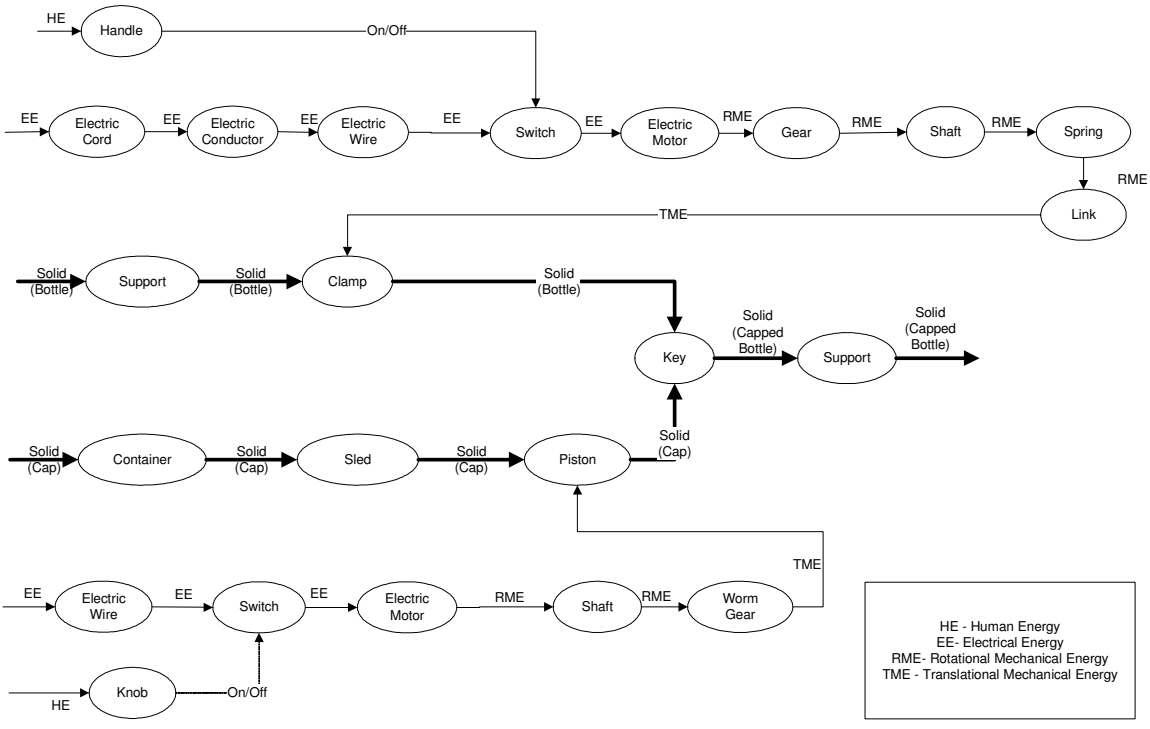


Figure 8.3 A design alternative created by the process using the grammar

To analyze the generated solutions, three major modules are defined that characterize the bottle capping machine design: the *cap module* that imports and places the cap into the device, the *bottle module* that positions and secures the bottle within the device, and the *coupling module* where the cap and the bottle are coupled to each other.

The first solution (Figure 8.3) includes a number of interesting component selections and their configuration that form these three modules. For the cap module, it is suggested that a container and a sled pair be used which is followed by a piston mechanism that carries the cap to the desired position within the device. The former of these concepts is leveraged from an iced tea maker design, where the tea leaves to be brewed were contained in a holder and then guided into a bigger opening via a sled. The latter is borrowed from a salad shooter design, where the user utilizes a hand driven piston to force the salad ingredients into a container before they are processed. For the bottle module, the generated solution includes a clamp mechanism which is inspired from a jar opener design. Finally, the idea suggested for the coupling module is to use a key feature. This principle of using a key to couple two solid objects was learned from a power screwdriver design where different types of screwdriver bits were keyed into a main shaft depending on the application.

The second solution (Figure 8.4) shows different ways of realizing the same functional requirements. In this design, a belt-pulley mechanism is suggested for the cap module. A similar concept was used in a single use camera product for advancing the film. For the bottle module, the design includes a spring loaded sprocket and a clamp mechanism. This complex group of components is borrowed from a jar opener design. Finally, for the coupling module the use of a punch is suggested. This idea is analogous to how a staple is punched into a paper in an electric stapler.

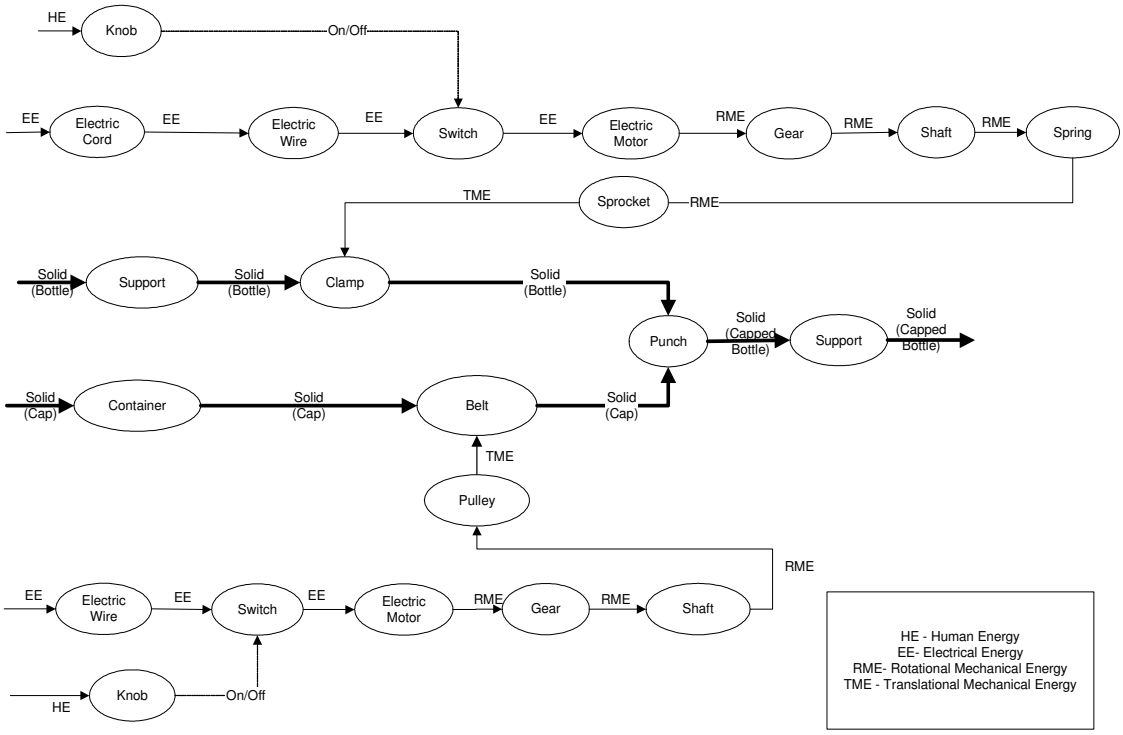


Figure 8.4 A second design alternative created by the process using the grammar

These examples demonstrate the richness of the alternatives created by the conceptual design grammar. In the next section, the results of the evaluation process for the bottle capping machine design are presented.

8.1.1 Finding Best Designs Using the Designer Preference Modeler

The goal of the evaluation scheme is to assess the worth of the generated design alternatives. As shown in Figure 8.2, the candidate pool for this problem includes 339,168 designs automatically generated by a total of 42 grammar rules. DPM, through its sampling algorithm selects 12 designs from this candidate pool. These candidates are added to the DFS and are presented to the designer for evaluation feedback. The designer evaluates the 12 candidates based on a single criterion of quality of concepts. The result

of this evaluation process is summarized in Figure 8.5, and the best candidate determined by the evaluation is shown in Figure 8.6.

Design Problem			
Example Problem	*Bottle Capping Machine*		
Evaluation Criteria	Quality		
Design Generation and Sampling			
# of candidates generated	**339,168**		
# of rules in the recipe pool	42		
# of candidates in DFS	12		
Design Evaluation		***Recipe***	***Score***
Best Candidate	C155,301	6,6,15,15,10,11,23,26,22,22,29, 39,39,44,47,50,53,55,57	**110**
Worst Candidate	C47	1,1,8,9,9,11,11,13,13,17,17,24, 39,39,44,48,51,56,58	**-112**

Figure 8.5 The tabulated evaluation results for the bottle capping design problem

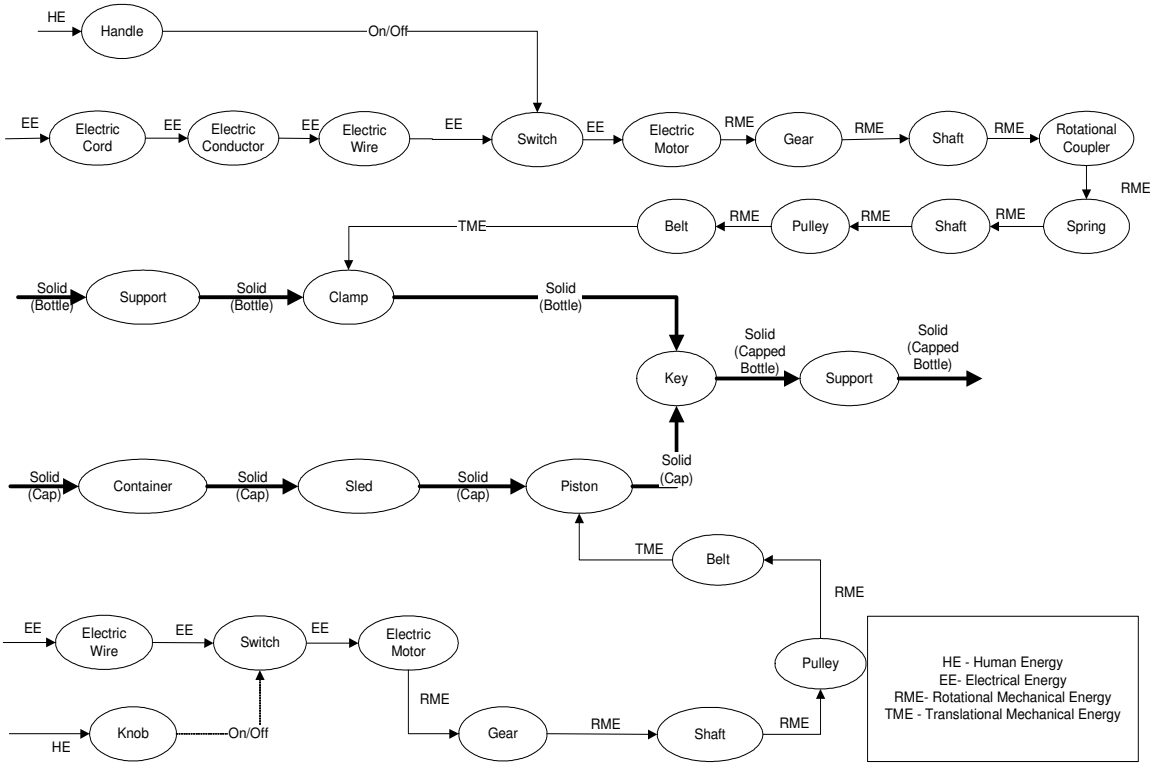


Figure 8.6 The best candidate in the population for the bottle capping design problem

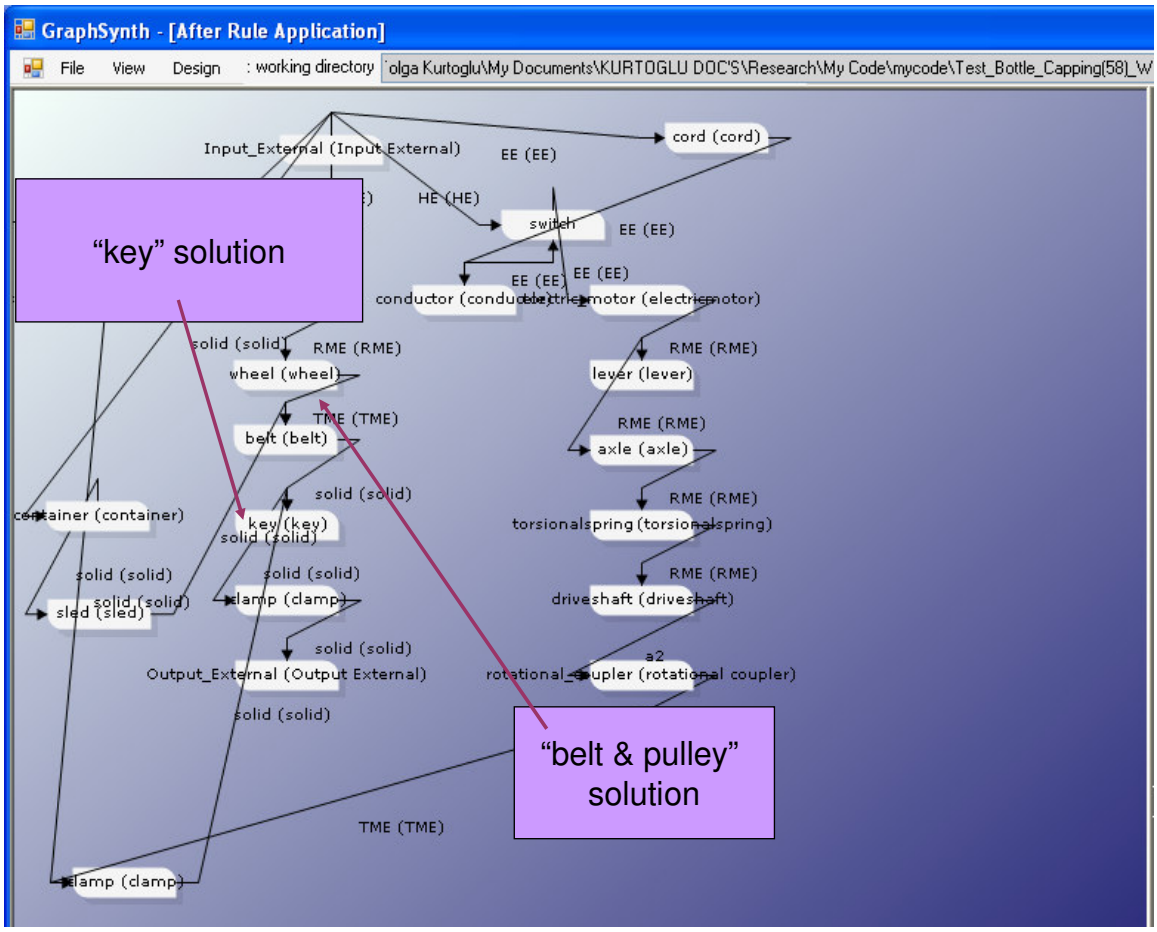


Figure 8.7 One of the candidates evaluated by the designer

There are several unique features of the best design. During the evaluations, the designer rated candidates involving the "belt and pulley" system (as a power transmission solution) very highly. Figure 8.7 shows one of the twelve concepts evaluated by the designer that includes a belt and pulley pair. These two components are inserted to the configuration by the program by invoking grammar rules #22 and #39. At the end of the designer evaluations, the calculated preference score of these rules are among the highest. As a result, the program searches and finds a "best" solution that uses a belt-and-pulley mechanism both for actuating the clamp of the bottle module and for driving the piston of the cap module, i.e. a candidate that includes two instances of rules #22 and #39. Note

that, however, none of twelve designs reviewed by the designer including the one shown in Figure 8.7 has a belt-pulley system for both modules. This shows the effectiveness of the DPM algorithm in understanding the designer preferences and in searching for designs in the population which are generated by the application of the rules that are most preferred by the designer.

Another example of this behavior is the "keying" solution. Similarly, the designer preferred candidates including the "keying" solution (such as the one shown in Figure 8.7) over others for the coupling module (other solution principles include using a punch or a hinge). As a result, rule #50 which inserts a key to the design has a higher preference score compared to both rule #52 which is used for a hinge solution and rule #71 which adds a punch to the design. This preference is also reflected on the suggested best design of Figure 8.6 which includes the component "key" in the final configuration and the rule #50 in its recipe.

8.2 DESIGN PROBLEM II: SODA MAKING MACHINE

The second problem is to design a device that takes water, sodium bicarbonate (gas), and soda flavor syrup as input and mixes them into a soda drink. In order to pose the problem, the designer supplies the conceptualized function structure shown in Figure 8.8.

This function structure also includes certain assumptions and design decisions. First of all, the device is targeted as a home type kitchen appliance. Therefore, the inputting of the water is assumed to be available through a standard kitchen faucet. It is also assumed that the soda flavor syrup is accessible in a separate container that can be poured into the device, and the sodium bicarbonate is contained in a canister that can safely transfer sodium bicarbonate into the system.

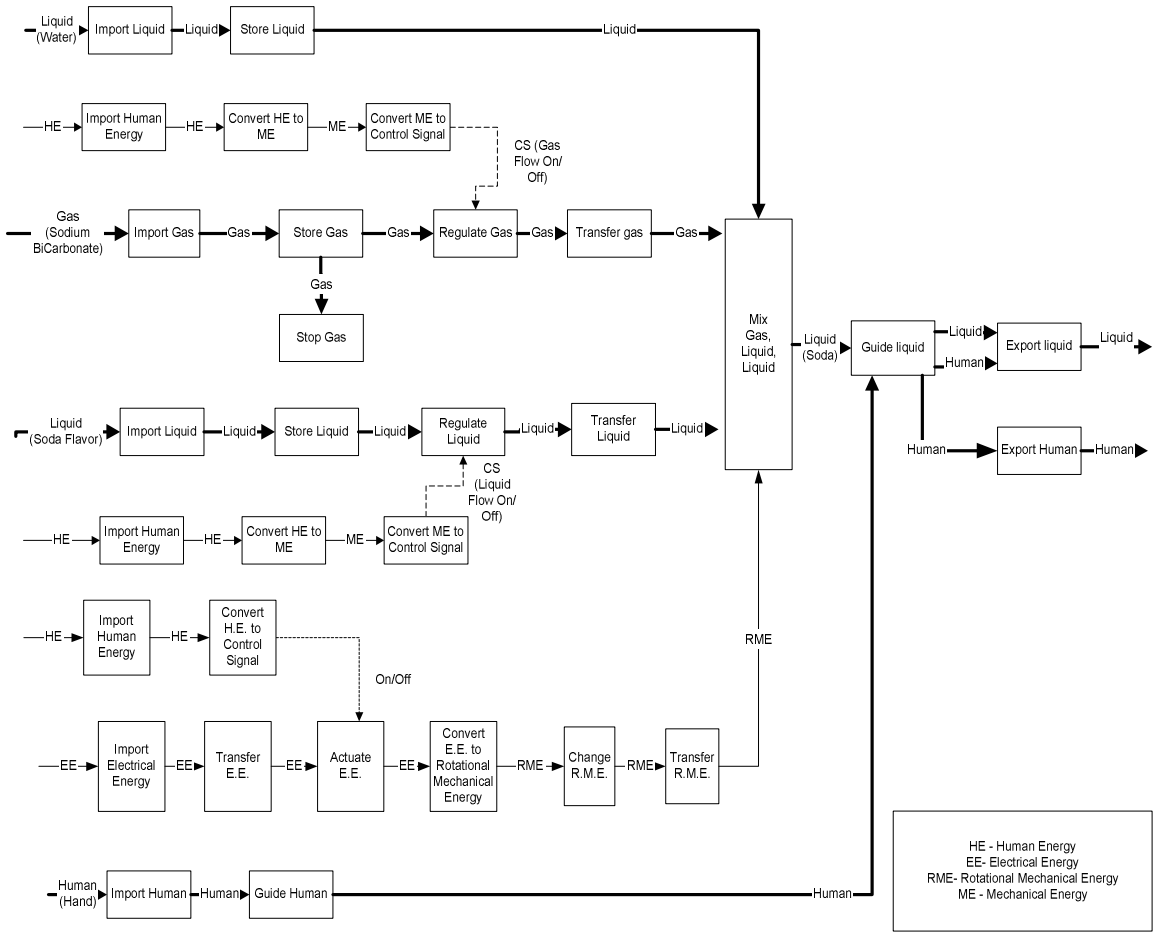


Figure 8.8 The function structure for the soda making machine

For this problem, the grammar generates 7,128 unique design configurations for the 31 sub-functions, and the 41 flows of the given function structure. The generation results are tabulated in Figure 8.8. To create the candidates for this design, 43 rules of the grammar database were invoked by the program. The resulting candidate set includes configurations consisting of 25 different component concepts. These components are listed in Figure 8.9. Two designs generated by the grammar are shown in Figures 8.10 and 8.12.

Design Problem		***List of Components***
Example Problem	*Soda Maker Machine*	conductor, switch, wire, cord, electric motor, knob, handle, gear, shaft, agitator, rotational coupler, sprocket, support, lever, link, clamp, worm gear, nozzle, tube, reservoir, valve, filter, seal, cover
# of Subfunctions in Input	31	
# of Flows in Input	41	
Design Generation Details		
# of different grammar rules invoked	43	
total # of grammar rules invoked	45	
# of candidates generated	**7,128**	
# of different components used in synthesis	25	

Figure 8.9 The generation results for the soda making machine design

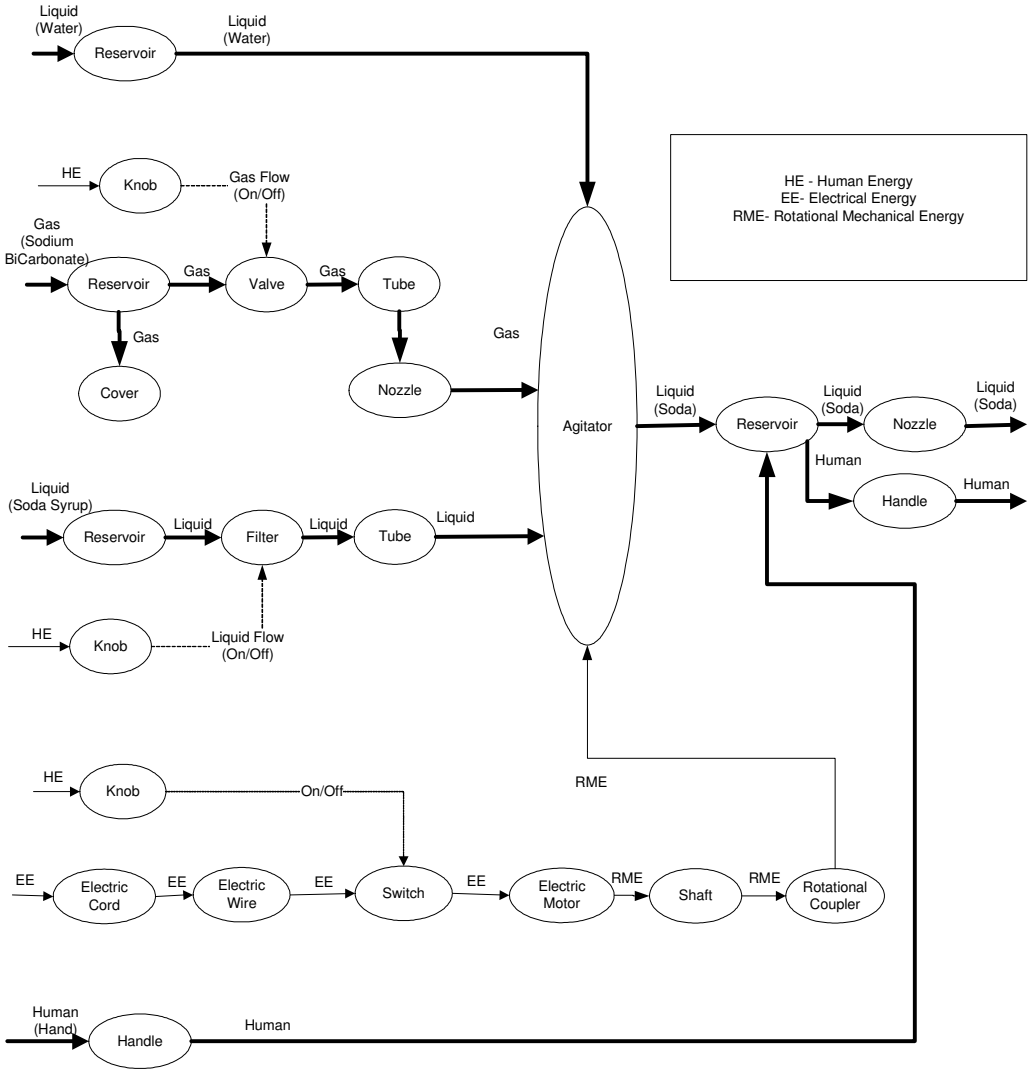


Figure 8.10 A design alternative created by the process using the grammar

The first solution (Figure 8.10) configures reservoirs, tubes, and valves and uses an agitator (stirrer) to mix the soda drink. The mixing solution is borrowed from the stir chef product. The rest of the aforementioned components are inherited from an iced tea maker, a snow cone machine, and a bug vacuum.

The variety of design solutions (and the total number of solutions generated) for this problem is much less compared to the bottle capping design. This is due to the fact that there are only a few solution alternatives to those functions that are not associated with transmitting mechanical power. For example, reservoirs and tubes are the only alternatives in the rule base to "import liquid", "store liquid", "import gas", store gas", "transfer liquid", and "transfer gas" functions. Similarly, for "regulating gas", the only rules associated include the component solutions valve and filter. This observation suggests that for certain functional requirements such as mechanical power transmission there are a greater number of standardized solution principles applicable.

Design Problem			
Example Problem	*Soda Maker Machine*		
Evaluation Criteria	Quality		
Design Generation and Sampling			
# of candidates generated	**7,128**		
# of rules in the recipe pool	43		
# of candidates in DFS	12		
Design Evaluation		***Recipe***	***Score***
Best Candidate	C5111	10,11,22,29,39,15,23,59,60,61,62,63,64,59,66,68,69,72,71,70,60	**119**
Worst Candidate	C498	1,11,13,39,8,17,59,60,61,62, 63,65,59,74,69,73,60	**-97**

Figure 8.11 The tabulated evaluation results for the soda making design problem

8.2.1 Finding Best Designs Using the Designer Preference Modeler

The best soda maker design is shown in Figure 8.12. The result of the evaluation process is summarized in Figure 8.11. Once again for this example, the best solution reflects the designer's preferences. Among the designer's favored components were seal

for the "stop gas" function (as opposed to cover), and valve for the "regulate liquid" function (as opposed to filter). Both of these preferences are represented in the design suggested as the best candidate.

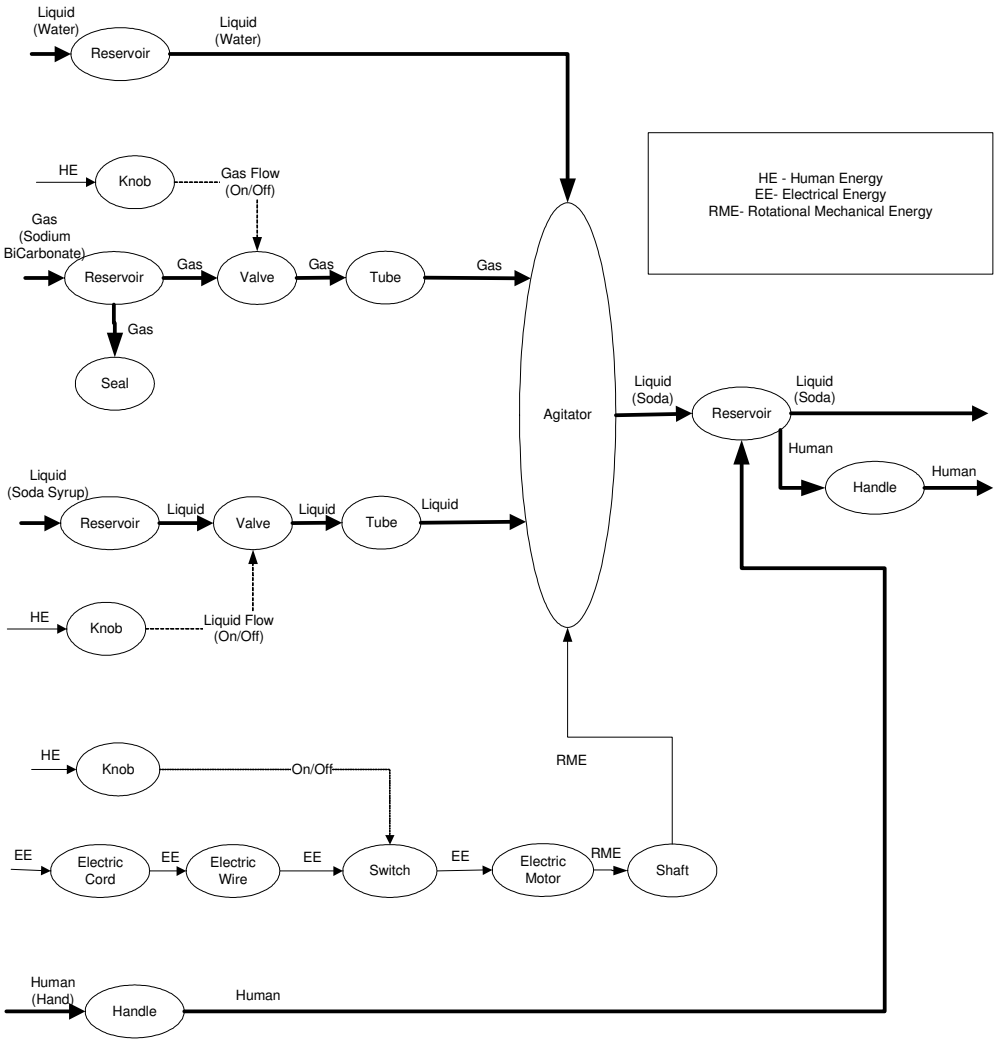


Figure 8.12 The best candidate in the population for the soda making design problem

8.3 DISCUSSION OF RESULTS

The two design problems demonstrate how a multitude of feasible conceptual solutions can be created by the use of the grammar and the modified BFS search that

invokes the rules. By utilizing an algorithm that systematically searches the design solution space, the grammar generates a rich set of design alternatives which are composed of solution principles that are successfully used in the design of past products.

An important question that follows is the validity of the solutions generated by the grammar. As a preliminary validation study, configurations of some existing designs are compared with that of the ones generated by the design automation tool. Figure 8.13 shows pictures of three existing "bottle capping" designs. (Picture 1 [accessed from http://www.aquatechnology.net/step7.jpg] and Picture 2 accessed from [http://www.bottlecapping.com/cappers/capper-benchtop.jpg] are found through an image search on the web. Picture 3 was taken by a colleague.) All three existing designs include components and solution principles that are captured by the grammar. These component concepts include container, sled, piston, knob, switch, support, belt, pulley, and punch. More importantly, the configuration of the components in these designs can be generated by the execution of grammar rules. The cap module of Design 1 of Figure 8.13, for example, is configured the same way as the conceptual configuration shown in Figure 8.3. Accordingly, a cap loaded in a container, is guided through a sled and positioned via a piston before it is coupled with a bottle. Similarly, Design 2 includes a belt-pulley system that translates the caps. This same solution was suggested by the automated design tool in the design labeled as "best" and presented in Figure 8.6.

These examples provide evidence that capturing solution principles based on their relations with functional needs is an efficient way of representing and storing past design knowledge. The grammar provides a framework where this past knowledge can be effectively used to create configurations that are comparable to existing designs. Moreover, unlike many expert system approaches to design automation, the computational tool developed in this research that can guide innovation by building on

past design knowledge in order to create entirely new concepts. The compatibility strategy described in Chapter 6 facilitates the formation of novel connections between components that can lead to innovative designs. Further discussion of validation and novelty are provided in Chapter 9.

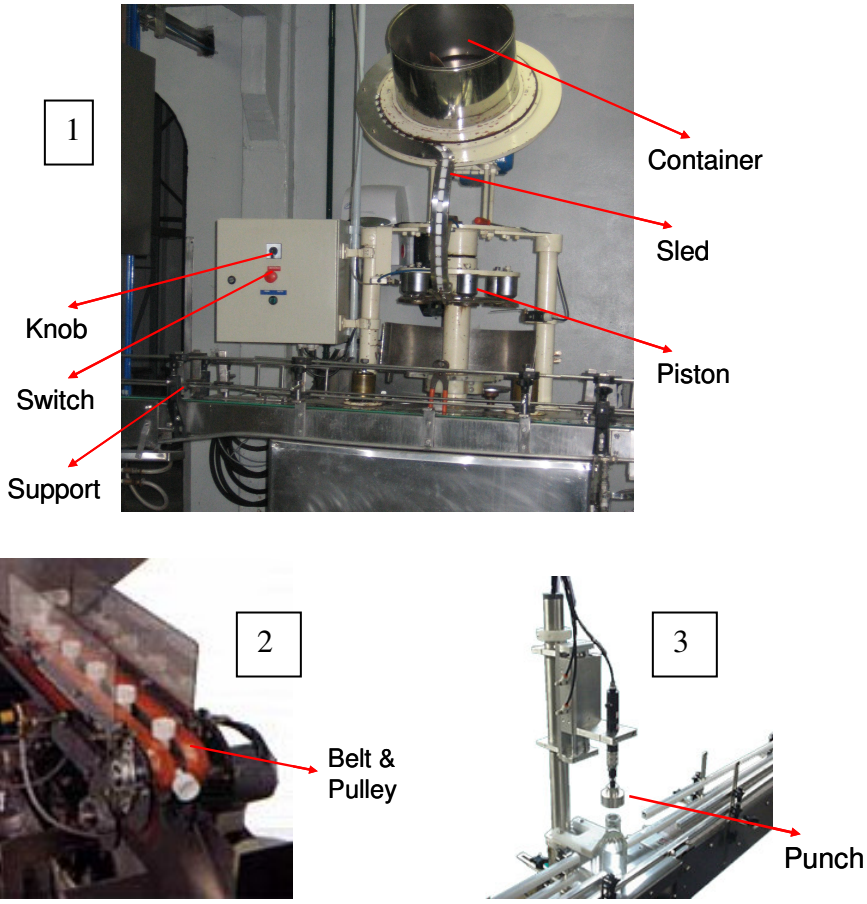


Figure 8.13 Three existing solutions to the bottle capping design problem

Another insight gained is that the DPM approach effectively samples from the solution space to find designs that are to be presented to the designer. The outcome of the sampling algorithm depends on the variability of the solution space. The more commonality there exists between generated candidates, the fewer the number of designs

that need to be evaluated by the designer. For the bottle capping design, 42 rules in the recipe pool are represented by 12 candidates. Similarly, for the soda maker design, 43 rules in the recipe pool are captured again by 12 designs. Considering the size of the solution spaces for these problems, the gain that results from evaluating only a dozen designs is considered to be well worth the effort. For the bottle capping problem, the designer feedback extracted from the evaluation of 12 candidates is used to find a best design among 339,169 candidates. As is evident from this example, the DPM approach overcomes the challenges common to many computational approaches to design generation - that is the creation of too many designs without a sound method for design evaluation.

Other observations include the fact that the candidates generated depend on the formulation of the input function structure. As expected, different designers may describe the functionality of the same product differently. This may still be the case even if the designers share a common vocabulary to model product functionality. The success of any computational tool then becomes reliant on the level of adaptability to different functional blocks used in a model. The grammar approach developed is highly responsive to topological variability in function structures. For example, if the “store RME” and “supply RME” functions are replaced with an “actuate RME” function in the function structure of the bottle capping machine, the grammar immediately responds by replacing the suggested “torsional spring” solution with a “latch release” without changing much else in the configuration.

There are also several areas where the presented grammar approach can be improved upon. First, the use of function structures and configuration flow graphs as the underlying modeling scheme for representing a system introduces inherent limitations. For example, non-functional, spatial, and temporal interfaces cannot be modeled using

these two graphical representations and hence are not currently accommodated by the presented framework. Second, the DPM method is limited to a single design objective or criteria. In conceptual design problems, there usually are multiple objectives that need to be handled simultaneously. Moreover, neither the design generation nor the design evaluations account for design constraints that may be present during conceptual design. For example, a certain connection between two components may be entirely feasible but unacceptable because of a design constraint. However, since the only evaluation comes from a real designer, the approach relies on that as a total evaluation of the design (e.g. cost, performance, reliability, etc.). Third, with the current approach it is hard to predict the final layout of the configuration. In other words, the CFG does not completely define how different sub solutions should spatially be arranged to arrive at the final overall solution. Finally, the completeness of the generated solutions depends on the knowledge stored in the grammar rule base. Sub-functions for which no grammar rules are defined cannot be mapped to design solutions.

Despite these limitations, the results of this chapter show that the implemented conceptual design grammar is able to generate solutions to open ended design problems by intelligently selecting and configuring components to address functional design requirements.

Chapter 9: Experimental Results

Conceptual design is at the very core of engineering and product development. While this fact is well recognized and acknowledged it is unclear how an automated design tool would benefit the concept generation process. It can be argued that the success of conceptual design heavily depends on the experiences or creativity of the designer involved. The more experiences or past design knowledge available to the designer, the more variability will exist in the concepts that are generated. On the other hand, more experiences could reduce creativity in that the psychological bias of past design knowledge would prevent a designer from finding truly creative solutions.

In this chapter, the results of some design experiments are presented. The experiments are conducted to understand how the grammar approach benefits the designers during concept generation. Accordingly, the computational method is tested using the two design scenarios presented in Chapter 8. The experimental approach involves 16 test subjects each creating as many concepts to two separate design problems: one in which outputs from the computational method was used, and one in which it was not. Three judges evaluated the resulting designs according to the metrics of variety, novelty and completeness. In the following sections, a presentation of the experimental details is provided.

9.1 EXPERIMENTAL METHOD

The experiments are developed to assess the effects of using the design configuration graphs as a design aid during concept generation. The fundamental research issue addressed here is to understand whether the configuration graphs benefit designers to create better ideas and thus better concepts. More specifically, the experiments seek to gain insight into the following research questions:

How useful are configuration flow graphs in creating design concepts?

To what extend can the variety of generated solutions be improved by the use of configuration flow graphs?

Does the use of configuration flow graphs produce more complete concepts?

Is there an interaction between the novelty of ideas generated and the use of configuration flow graphs as a design aid?

9.1.1 Design Problem Descriptions

The two problems introduced in Chapter 8 are used as the test cases for the experimental studies. For each problem, the conceptual function structure of the product (Figure 8.1 and 8.7) is given to the designers along with the problem description.

9.1.2 Designers

The participants are graduate students from the University of Texas at Austin. These students, a total of sixteen, average 12.8 months of professional engineering experience (out of school) that ranges from no experience (3 participants) to 4 years of experience. They all have recently studied in a graduate design methodology course that teaches a number of idea generation techniques including Brainstorming, Mind Mapping, 6-3-5, patent searching, TIPS, and design by analogy. In the same course, they also learn about how to create function structures using the functional basis, thus the participants are proficient in reading and interpreting function structures such as the ones shown in previous chapters.

9.1.3 Experimental Procedure

The main goal of the experiments is to assess the value of the configuration flow graphs as a conceptual design aid. To explore the effect of the use of configuration flow graphs, a series of experiments are conducted. Accordingly, a set of sixteen participants is

randomly divided into four experimental condition groups. Each group includes 4 designers who *individually* generate concepts to the two design problems. For each individual, one of the design problems is accompanied by a set of configuration graphs that is presented as catalog-like reference material that the participant may choose to use as a design aid for concept generation. For the other problem, no design configuration graph is provided. The order of design problems is also altered for different sets of individuals.

The design configuration graphs that are given to the participants are automatically created by the computational synthesis tool. A set of CFG's are then computed for each problem following the sampling algorithm described in Section 7.4.1. In order not to bias the experiment, no concept scoring/ranking is performed using these sets. The sampling algorithm resulted in 12 CFG's for each problem. Among these, 6 are randomly selected for the bottle capping design, and three are selected for the soda making design. These CFG's are then redrawn in Microsoft Visio and presented to the designers as part of the experiment.

The experiment is conducted over a two week period. The participants are asked if they have participated in any form of concept generation exercise for the given problems prior to the session and are excluded from the experiment if they have done so. At the beginning of each session, the experimenter read a script of instructions explaining the experimental procedure. The instructions include a description of the problem, a conceptual function structure of the problem, and basic guidelines for the prescribed method. The guidelines outline that the participants are expected to generate as many solutions in the allotted time. The participants are also told that they should represent their ideas using sketches and/or written words and that their generated concepts should address majority of the functional requirements specified by the given function structure.

For sessions including CFG's as a design aid, there are additional instructions explaining what a configuration graph represents followed by the selected set of CFG's for the particular concept generation session. In these sessions, participants are asked to study this additional material before generating concepts and are instructed that they may use concepts from the presented configuration graphs during their exercise. The sessions last approximately 50-60 minutes with an equal 45 minutes allocated to idea generation whether or not the session includes the use of CFG's. Each session is followed by a post-session questionnaire that asks questions about the experiment and the use of configuration graphs.

9.1.4 Evaluation Metrics

The development of proper metrics for concept evaluation during engineering problem solving is an emerging field of study (Otto and Wood, 200; Otto and Holtta, 2004). Shah et al. (2000) proposes a set of four metrics specifically developed for the evaluation of idea generation methods used for conceptual design. These four metrics are defined based on how well a concept generation method expands a design space, and how well it explores the resulting space.

In the evaluation of experiment results, two of these four metrics (novelty, and variety) are used and a new one (completeness) is added. The definitions of these metrics are as follows:

Completeness: is a measure of the level indicating how much a concept variant addresses the sub-functions depicted in the function structure.

Novelty: is a measure of how unusual or unexpected an idea is as compared to other ideas (Shah et al., 2000).

Variety: is a measure of the explored solution space during idea generation (Shah et al., 2000).

9.1.5 Evaluation Procedure

Two doctoral candidates and a faculty member are selected as judges from the Manufacturing & Design Division of the Department of Mechanical Engineering at the University of Texas at Austin. All three evaluators are experts in the areas of functional modeling and concept generation and all have given lectures on related subject matters. The judges independently evaluated the resulting concepts according to the metrics of completeness, novelty, and variety. The first two of these metrics are evaluated individually for each concept generated, whereas the last metric is evaluated on a set basis for each participant.

To perform the evaluations, all three judges are given a shuffled stack of concept sketches without knowing which designs were created with the aid of the configuration flow graphs and which without. Each judge is instructed to go through all solutions and evaluate them according to the guidelines summarized.

For concept scoring, an ordinal 1 (low) to 7 (high) scale is used. Before running statistical analysis on the evaluation data, final scores are computed by averaging individual scores of the three judges. Each metric are then averaged for each participant. To monitor agreement between judges, pair wise Spearman rank-order correlation coefficients (Shah et al., 2003) are also calculated. For the soda maker problem an acceptable level of correlation occurred for all metrics: completeness [0.52-0.57], novelty [0.44-0.63], and variety [0.44-0.78]. For the bottle capping device, on the other hand, only the completeness metric yield adequate correlation [0.50-0.75]. The other two metrics for this problem did not have good correlation between the judges and are therefore excluded from further analysis. This also suggests that more investigation is necessary to understand why the judges were in disagreement in their variety and novelty scores for the bottle capping problem.

9.2 EXPERIMENT RESULT

A total of 125 (56 for "Bottle Capping Device", and 69 for "Soda Making Device") conceptual sketches were created by the participants with varying degrees of completeness, novelty and variety. Six of these sketches are presented in Figure 9.1 and 9.2. From an initial look to the overall outcome of the experiment, it is obvious that the participants have devoted their time to the two design problems to arrive at high quality, innovative solutions.

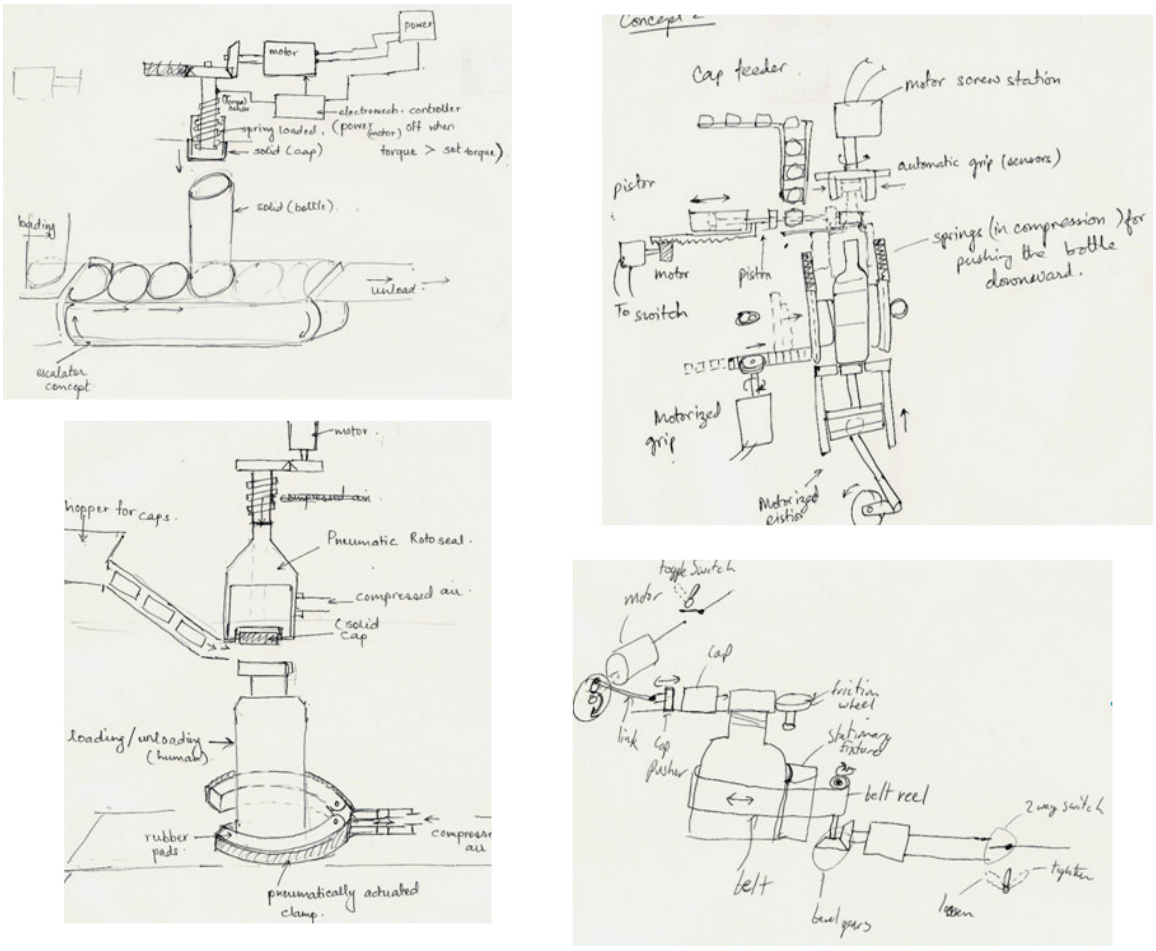


Figure 9.1 Sample sketches created by the participants for the bottle capping design

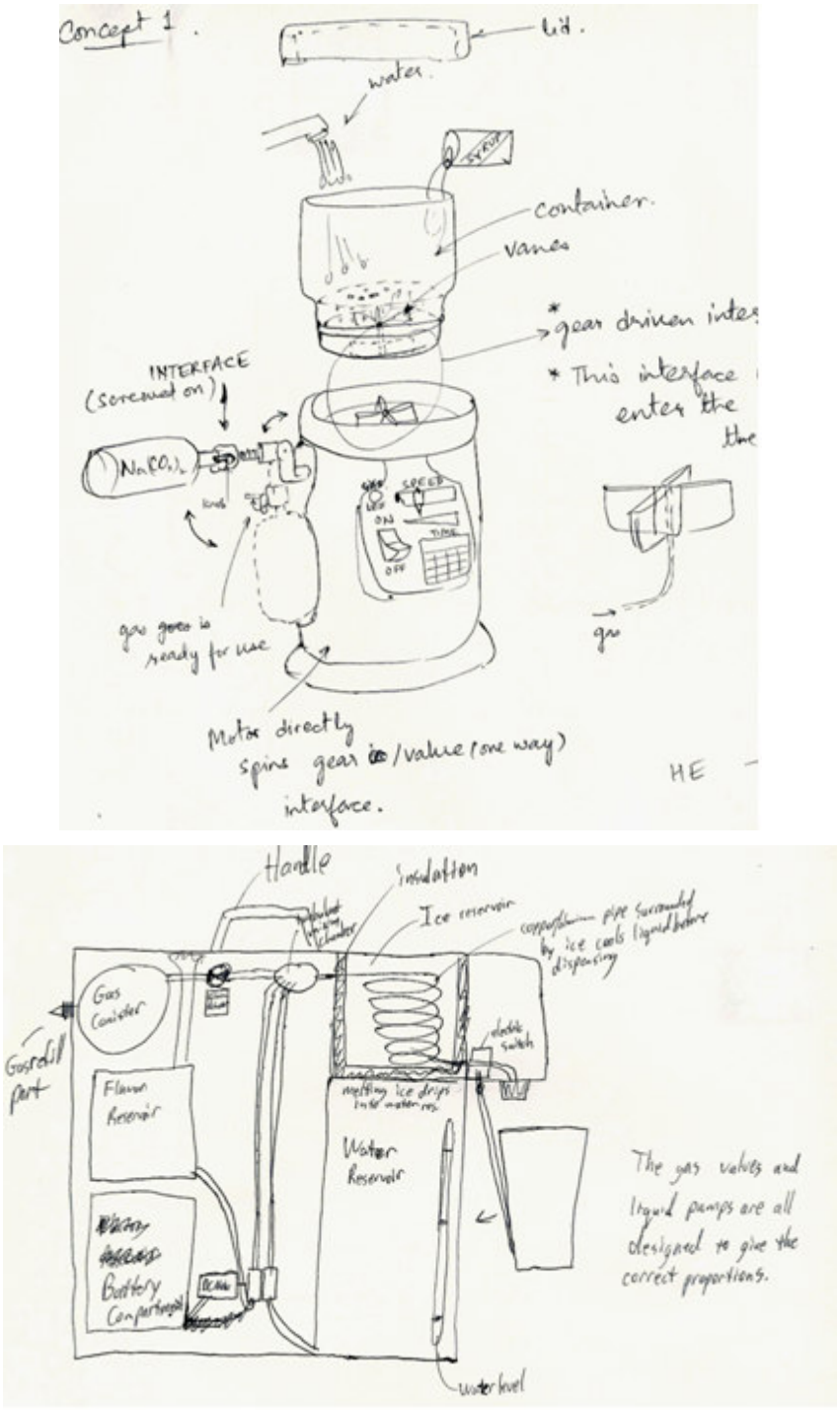


Figure 9.2 Sample sketches created by the participants for the soda maker design

First, the analysis starts by evaluating interdependency of the chosen evaluation metrics. To verify that the metrics are independent, the Spearman correlation coefficients between the three metrics are calculated (Figure 9.3 – for Soda Maker). The results show that there is no correlation between the metrics for the individual concepts thus indicating that the metrics are independent. Interestingly, there is correlation between a concept set's average novelty and its variety. This does not imply the metrics are correlated instead it implies concept sets containing very novel ideas tend to have greater variety overall. This is not surprising since novel ideas tend to be very different from common ideas and it is easy to think of the common ideas. The concept sets with a high degree of variety are likely to contain a combination of very novel ideas and very common ones.

	Spearman Correlation Coefficient		
	Novelty and Completeness	**Novelty and Variety**	**Completeness and Variety**
Judge 1	0.12	0.25	0.26
Judge 2	0.28	0.49	0.1
Judge 3	0.14	0.36	0.13
Ave. of 3 Judges	0.18	0.77	0.16

Figure 9.3 Spearman correlation coefficients between evaluation metrics

Secondly, the problem order effects are analyzed. Even though the problem order was counter-balanced in the experimental setup, it is required to verify that problem ordering did not have any effect on the outcome. To achieve this, the data is analyzed using a three-factor ANOVA with repeated measures on the 2nd and 3rd factors and a Mann-Whitney U test. (Initially ANOVA was used to analyze ordering effects, even though the data deviates from this test's assumptions. There is not a common approach for analyzing ordering effects of ordinal, non-normally distributed data (Gliner and Morgan, 2000). According to the results, there is no significant effect on completeness due ($F=0.036$, $p=0.857$; $U=124$, $p=0.897$) to the order in which the problems are

presented to participants. After verifying that, the data is collapsed for the rest of the analysis.

To continue the analysis, the data is normalized for each participant by averaging the individual scores produced by a given participant. This ensures that the analysis is not biased towards participants who create greater number of concepts. Participants produced anywhere from two to seven concepts for the soda maker problem and one to six concepts for the bottle capping problem. This data is presented in, Figures 9.4 - 9.7.

		Soda Maker			Bottle Capping
		Completeness	Novelty	Variety	Completeness
with CFG	25 percentile	3.81	3.6	3.83	2.44
	Median	4.17	3.74	4.5	3.92
	75 percentile	5.17	4.17	5.08	4.25
without CFG	25 percentile	3.31	2.87	2.58	3.13
	Median	3.53	3.59	3.5	3.54
	75 percentile	4.2	4.24	4.33	4.18

Figure 9.4 Distribution of experimental data

Next, the statistical analysis on the normalized data is run. In addition to this initial, combined analysis, the top two ideas, as determined by the sum of the completeness and novelty scores, from each participant are also analyzed. For the bottle capping device, the top two ideas are not analyzed since one participant produced only a single concept. All the statistical analyses are performed using Mann-Whitney U tests (David Clark-Carter, 2000) running on SPSS statistical analysis software. Mann-Whitney U is used rather than t-test, ANOVA or MANOVA because the data is ordinal and the judges scores are not close to being normally distributed. Mann-Whitney U is the non-parametric equivalent of a t-test and compares the ranked scores of the two groups.

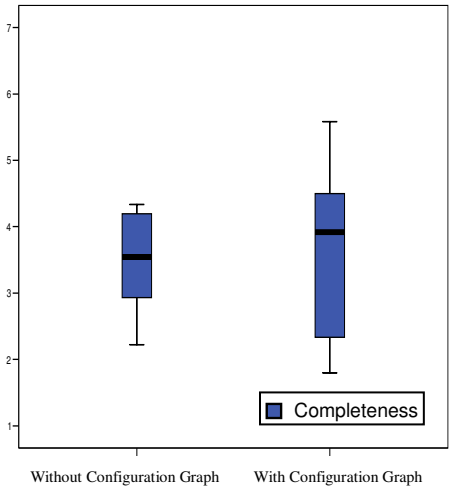


Figure 9.5 Box plot summary for the bottle capping device

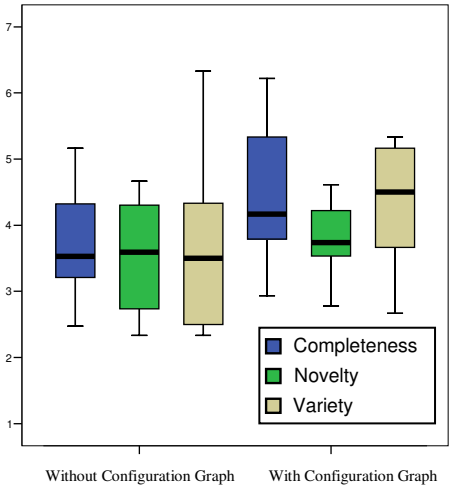


Figure 9.6 Box plot summary for the soda maker device

Analysis results are summarized in Figures 9.8 - 9.10. In short, the analysis results show the configuration flow graphs having a potential to enhance the idea generation process meriting further development and study for their use during concept generation. The best ideas from each participant for the soda maker problem are statistically more complete when participants have access to configuration flow graphs than when they do

not (Figure 9.9 & 9.10). More importantly, while configuration flow graphs show the potential to support the design process by helping the designers develop more complete designs, they do not hinder the novelty or the variety of ideas produced. Even when only the most novel concepts from each person are analyzed there is no difference between the groups (Figure 9.11).

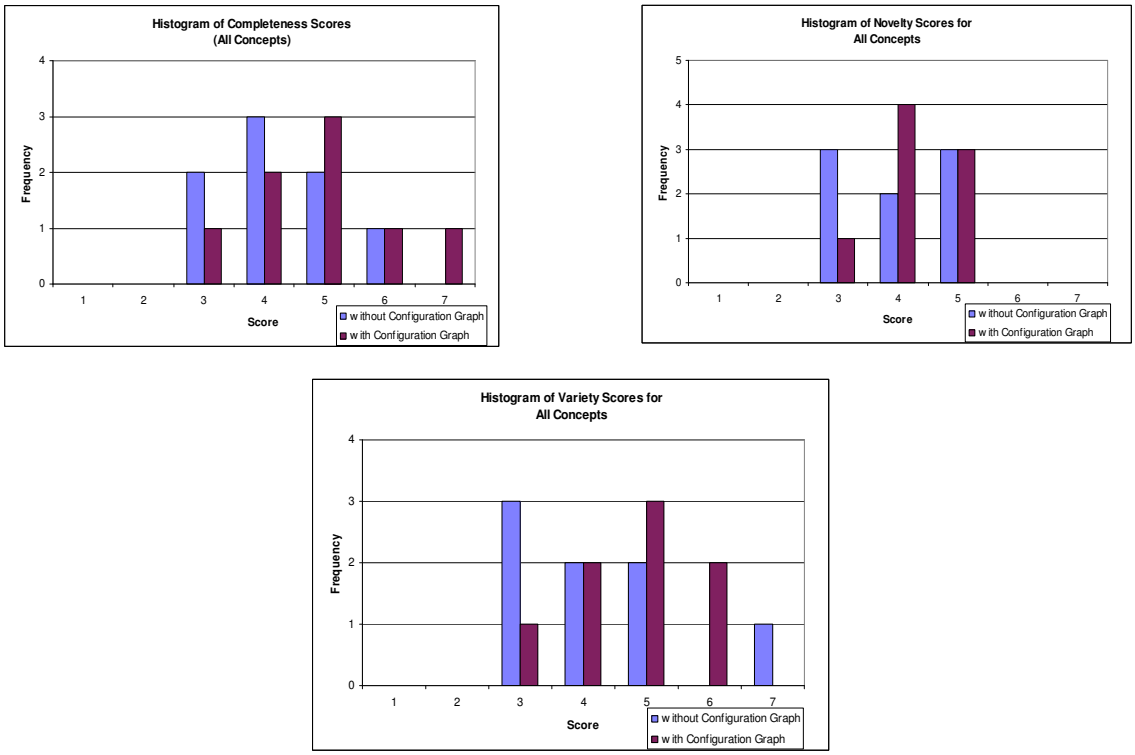


Figure 9.7 Histograms showing the score distributions for the soda maker design

	Soda Maker			Bottle Capping Device
	Completeness	Novelty	Variety	Completeness
Mann-Whitney U	19	28	19	31
p (significance)	0.19	0.72	0.19	0.96

Figure 9.8 Mann Whitney U test results for normalized data

	Soda Maker		Bottle Capping Device
	Completeness	Novelty	Completeness
Mann-Whitney U	14.5	27	28.0-31.5
p (significance)	0.07*	0.65	0.72-0.96

*statistically significant

Figure 9.9 Mann Whitney U test results for the best idea from each participant

	Soda Maker	
	Completeness	Novelty
Mann-Whitney U	77.50	93.00
p (significance)	0.06*	0.20

*statistically significant

Figure 9.10 Mann Whitney U test results for top two ideas from each participant

	Soda Maker	
	Completeness	Novelty
Mann-Whitney U	17.00	19.00
p (significance)	0.13	0.20

Figure 9.11 Mann Whitney U test results for the most novel idea from each participant

9.2.1 Observations

To better understand the usefulness of providing designers with configuration flow graphs, a post-session questionnaire asked the participants several questions concerning how they felt about using the graphs. The answers to two such questions are summarized in Figure 9.12. It is easily observed that the designers liked the option of being provided the configuration graphs. This was more the case for novice designers with less professional experience.

From an open-ended question on the survey that stated, “Please make any additional comments you have about the concept generation exercise”, numerous participants reiterated the fact that they found the configuration graphs to be beneficial. Some comments from the participants included: “the design configuration graphs were

helpful…", "the design configuration graphs give you some place to start…", "the configuration graphs definitely helped, and the solutions generated by the configuration graphs were very exhaustive and comprehensive." In addition, some students complained about being constrained or fixated by the presence of the design configuration graphs. This connotation and the overall experiment results are discussed in the next section.

Survey Results from Concept Generation Sessions (16 Participants)						
Question	**Very Difficult**	**Difficult**	**Somewhat Difficult**	**Somewhat Easy**	**Easy**	**Very Easy**
How easy was it to understand the design configuration graphs?	0.00%	0.00%	6.25%	18.75%	68.75%	6.25%
Question	**Not Useful**	**Somewhat Useful**	**Useful**	**Very Useful**		
How useful was the design configuration graphs in creating concepts to the given design problem?	0.00%	37.50%	43.75%	18.75%		

Figure 9.12 Experiment survey

9.3 DISCUSSION OF EXPERIMENT RESULTS

In this chapter, the implications of integrating an automated design tool into the concept generation process are investigated. The aim was to provide designers with a tool that would benefit them during idea generation by presenting them knowledge that may fall outside their personal experiences (or immediate memory) and to enable them to readily reuse existing design knowledge. Providing designers with such design aids, however, do carry some risks. First, they may hinder creativity, and second, they may cause design fixation that prevents designers from exploring truly novel ideas.

A number of research questions are formulated to assess the effects of the automated conceptual design tool on concept generation. The results of the study provide initial evidence that the automated tool positively effects and, in fact, supports the human idea generation process. It is evident that the configuration flow graphs have the potential to help designers create more complete concepts early in the design process by allowing them to approach the problem in a more systematic, function-oriented way.

Moreover, contrary to the claim that stated the configuration flow graphs cause fixation on certain ideas during concept generation, no statistical evidence was found that suggests the use of the graphs hinders creativity or variety. Currently, the rule database consists of a mere 23 products so it would not have been surprising if the variety or creativity was limited due to the limited size of the database. However, even this relatively small database facilitates the development of concepts that are not dominated by traditional concept generation methods for novelty and variety. Therefore, the automated synthesis tool warrants further development and study as a supplement to human idea generation.

In addition, the survey results indicated that designers felt positive about using the configuration graphs. They found the graphs easy to understand and to their benefit in creating conceptual solutions. An open question is if a more complete concept early in the design process will lead to a better final product or a better, more efficient design process. Other design methods such as QFD (Houser and Clausing, 1998) and functional modeling also lead the designer to more completed and detailed designs early in the process.

This study was designed to provide preliminary evaluation of the configuration graphs and their possible usefulness in the design process. Necessarily, there have been a number of assumptions made that need to be taken into account when interpreting the results. The idea generation was an individual not a team activity. The short 45 minute time period for idea generation is a bit short for the complexity of the design problems but was long enough for many participants to run out of ideas. Also, this study only took a snapshot of the design process just before and just after the idea generation. This was done to set-up a reasonable experiment and to limit the noise that other phases in the

design process may incur. However, the effects of this decision on the quality of the final product are not known and needs to be studied.

Nevertheless, the development of the computational tool was intended to help designers choose suitable components for given functional requirements in a redesign or an original design situation. The data from the experiments support the hypothesis that the integration of automatically generated design aids such as the configuration flow graphs into the concept generation process results in an improved idea generation performance. The designs obtained by using configuration graphs proved to be more complete, without any decrease in variety or creativity.

Chapter 10: Conclusions

This chapter summarizes the research and provides a discussion of the contributions and potential venues for future work.

10.1 SUMMARY

In this research, a computational methodology for creating conceptual design configurations is introduced. The method is based on leveraging existing design knowledge which is captured through an empirical study that involves systematic dissection of various products and the construction of a design knowledge base. This design knowledge is integrated into a formal framework that enables the modeling of existing designs and the creation of new concepts.

The framework includes representations that capture designs at two domains (the functional domain and the component domain) and, it facilitates the formulation of a design grammar language that captures the mapping between them describing how electromechanical components addres functional requirements in existing designs. The grammar rules created from the empirical analysis are then included in a computational search process that works with a designer in navigating the design space to create conceptual design configurations from detailed specifications of product function. The configuration description includes the choice of components and their topological structure consisting of the connectivity of components and the physical interfaces that result.

In addition, a tool called the DPM (Designer Preference Modeler) is introduced that analyzes these automatically generated design configurations by gathering preference information from a designer. The DPM method establishes an interaction between the designer and the computational synthesis tool in order to extract a designer's decision

making and preferences during concept evaluation. This knowledge is then used to build a model of designer preferences to be used for assessing the worth of automatically generated design alternatives.

Overall, the developed suite of design tools (the grammar driven concept generator, and the DPM) can be used to generate multiple and feasible configurations, which then can automatically be evaluated and ranked based on user-defined design objectives. The implemented system serves as a comparison to human conceptual designing and as a tool to complement the skills of a designer or a design team.

10.2 DISCUSSION

As described in Chapter 6, the grammar help designers automatically create design configurations through a computational search process that successfully integrates component concepts from different products into multiple complete concept variants.

There are various advantages of the design generation approach developed in this research. First, the method is based on a *design reuse* philosophy. Design reuse plays a vital role in product development especially during the early idea generation phase where most of the cost of a product is committed (without much statistical proof the 80/20 rule is referenced in the literature, stating that 80% of the cost is determined by the decisions made within the 20% of a project) Therefore, it is important for most industries to adopt a design-by-reuse process for cost effective product development. The computational approach developed in this research directly supports design reuse in conceptual design.

Second, the search-based generation approach eliminates potential solution bias. Designers with varying levels of experience in different domains usually rely on solutions derived from domains where they have the most expertise. This may cause design fixation and prevent the designers from exploring truly novel solutions that fall outside their expertise memory. The grammar method searches the design space without any bias

by systematically applying each grammar rule with equal likelihood. As a result, a vast space of past design solutions is explored and design configurations are generated by combining solution principles from different engineering domains. Examples of this are provided with the two electromechanical test cases presented in Chapter 8.

Third, the feasibility of the resulting configurations is ensured. Since designs are built incrementally by selecting only compatible solution principles or components, the addition of physically incompatible components to the design is prevented within the grammar. This is unlike other computational methods where a large number of complete solutions is generated first and pruned later based a feasibility criteria.

Fourth, the "flow" based compatibility measure employed by the grammar provides a proper level of flexibility to allow the formation of novel connections between components. This is contrary to most bottom-up approaches to design synthesis that define connection feasibility more restrictively by either utilizing a prescribed set of ports and their physical types or by employing hard constraints that prohibit connections that are not seen in the past designs.

Finally, the open-endedness of the grammar formulation enables the synthesis of multiple-input multiple-output systems and accounts for function and structure sharing. Accordingly, multiple functions can be addressed by a single component or multiple components can address a single function. The notion of function and structure sharing is an important aspect of conceptual design that defines the conceptual architecture of a product, and should be supported by automated synthesis tools.

In addition to the conceptual design grammar, a tool called the DPM is developed to analyze the designer's decision making during concept evaluation, and to construct a designer preference model that can be used for evaluation of automatically generated design alternatives.

The main novelty of the DPM approach is the combination of computing and human reasoning. Computational design tools can generate numerous solutions to a design problem by conducting an in-depth search of the design space. But, they typically require very detailed specifications of part shapes and dimensions in order to evaluate certain performance parameters. At the conceptual stage of design, however, such detailed models of system components and design parameters are simply not available and the evaluation of conceptual design alternatives remains a stumbling block in computational synthesis. Humans, on the other hand, can employ sophisticated reasoning techniques in making decisions for ruling out or approving design alternatives, however they are bounded by the limits of their own rationality (Simon, 1969). This makes it impractical for them to process and evaluate more than a handful of alternatives. The heuristic followed by the DPM takes advantage of the commonality of solutions in a design space and ensures that the required number of designer evaluations is kept to a minimum, consistent with the principle of bounded rationality. By establishing a proper level of interaction between the designer and the synthesis software, DPM combines the strengths of computational search with that of human decision-making. Moreover, since the preference model is directly derived from the evaluations of the designer, it reflects the designer's reasoning and judgment under specific design objectives and constraints which is easily generalized and used for faster search towards the best designs in large design spaces as is shown for the designs of the bottle capping and the soda maker devices.

Despite these advantages, there remains a number of theoretical and practical challenges that may be addressed to further advance the automated synthesis method developed here. For example, the reverse engineering approach to knowledge acquisition brings a number of limitations. The quality and completeness of the resulting grammar

rules are dependent on the person performing the breakdown analysis and product modeling. Even though the contributors to the product dissection efforts first go through a training process before constructing functional and configurational models of the products, unavoidably there exist cases where different interpretations are valid for modeling decisions or where mistakes are made in deriving models and defining grammar rules. These can be kept to a minimum by holding design review discussions among all parties involved with the goal of standardizing the reverse engineering practices and different interpretations of functional and configurational design aspects. In general, the repeatability of product modeling and grammar rule derivation has been high.

A more interesting issue arises from the theoretical claim of the research that it attempts to extract the intent of the designer in mapping function to form. One may argue that, in fact, the knowledge acquisition scheme followed does not truly capture the designer's decisions or intent, and that the actual functional description used by the designer cannot be known. Thus, the generated rules only reflect the modelers understanding of the product under consideration, which may be different than the original intent of its designer. This is a fair claim, but the distinction is subtle. The outcome of the knowledge acquisition is believed to measure something that correlates strongly with designer intent and, in same cases, may actually be superior to it in different ways (e.g., something unintended may actually improve functionality in some way).

Another challenge is with that of the scaling of the computational implementation. This challenge is twofold. First, as the nodes in a functional graph of a product increases, so does the branching factor of the search tree. Similarly, increasing the number of rules in the database poses a computational burden on the search algorithm as it increases the number of applicable rules. This growth of computational requirement may be overcome

in one of three ways. First, the design space can be navigated using a different search technique. Stochastic approaches to search such as genetic algorithms or simulated annealing are more appropriate for searching larger, multi-modal spaces. Secondly, the problems that are larger in size can be decomposed into sub-modules and the search algorithm can be run on the defined modules. For example, if one is looking to design a solar car, the transmission, the body, the engine etc. can be functionally described separately as main modules of the design and the synthesis results can later be integrated into a whole. Of course, a separate set of rules need to be developed to handle module compatibility and integration issues. Third, the rule selection may be altered such that certain rules are preferred over others using a probabilistic technique. Such an approach can drastically reduce the number of rules that need to be invoked during the search for solutions.

Finally, there may be some limitations on the use of the developed computational tool stemming from the level of design training of the user. Because the synthesis approach requires the design problem to be cast as a functional model, it inherently assumes the designer to be sufficiently experienced to perform an efficient functional decomposition.

This discussion has presented the advantages and limitations of the developed computational synthesis method. Regardless of the presented challenges, the conceptual design grammar has opened the door to realizing the possibilities of automating the conceptual design process and mimicking some of the cognitive activities that govern human creativity and the ability to invent.

10.3 CONTRIBUTIONS

The foundations of this research are function-based synthesis, reverse engineering through product dissection, knowledge representation, graph grammars, and learning-

based search. As a result, the benefits of the research affect these fundamental research areas though to varying degrees.

At the highest level, the research activities outlined here significantly impact basic design theory research and applied design. By combining formal design synthesis methods with structured algorithms a formal computational theory for conceptual design is created and implemented. Main contributions of this research include the creations of synthesis focused solution knowledge, computational methods for producing a large range of candidate solutions, and evaluation algorithms that reduce the range of possible solutions to a tractable set of "best" solutions. Several specific contributions are identified as:

- The creation of a knowledge base of designer decisions for concept generation (the grammar rule set).
- The development of a formal computational method that automates the systematic function-based design synthesis process (Pahl and Beitz, 1988). The method is embodied in a design tool that is capable of searching infinite design spaces for component configurations to meet functional specifications.
- The categorization and classification of the electromechanical component space (component basis) that can be used for computational synthesis.
- The development of a graph-based design representation, namely the configuration flow graph (CFG), to be used with computational or traditional concept generation techniques.
- The development of an evaluation method to assess the worth of design concepts in lack of specific geometry information (design preference modeler).
- A research process that combines empirical and computational methods to develop knowledge-based tools for automating the design process

10.4 FUTURE WORK

The development of the conceptual design grammar and the resulting computational tool allows for future work to proceed in a variety of directions. These directions can be grouped under four major categories: knowledge representation and rule derivation, design generation, design evaluation, and level of automation. They are presented next.

10.4.1 Knowledge Representation and Rule Derivation

Designing a knowledge representation language that is sufficiently expressive to capture the wealth of design knowledge about complex devices is a common challenge. The current framework of the conceptual design grammar models function and configuration product knowledge. However, the representation scheme can be further developed to be able to represent other design aspects such as constraints (e.g. width < 3in; rust proof), preferences, priorities, and tradeoffs (e.g. low power is more important than lightweight), states of the device (e.g. "when threshold temperature is reached"), properties of device, its materials, and its use (e.g. deformability, breakability, environmental stresses, etc.) This expansion of design modeling can be used both for design generation and for design evaluation.

Related to this discussion, another future goal is to accommodate structural design specifications in addition to functional specifications. Function structures are widely accepted as a functional representation; however they are limited in their ability to represent structural aspects of design. Towards that goal, there is a need to develop a representation that would complement the configuration flow graphs (CFGs) and facilitate the addition of structural components and interfaces into the conceptual design configurations.

The automation of rule derivation is another interesting extension to the presented research. The set of 189 rules has been created through the careful deliberation that is typical of creating any grammar rule set. In the current approach, the flow information provided by the two graphs is utilized in order to extract the mappings between a functional model and a configuration flow graph. Accordingly, strict boundaries are defined on both graphs by utilizing the shared flow connectivity. These boundaries isolate the mapping between functional nodes of a function structure and component nodes of a configuration flow graph. This process can be codified by leveraging graph-theoretic algorithms that search for intersection of these two graphs. The result of such a search process can extract potential grammar rules automatically.

10.4.2 Design Generation

An improvement area for more intelligent design generation is the development of a selection strategy for the rules to be applied. Such a strategy will encompass rules or guidelines that would enable the computer to select among recognized rules based on the design context, and the current state that the design is in. In addition, these guidelines can be automatically extracted from past design evaluations and can be defined as “meta-rules”. This separate family of rules can be used to formulate a knowledge base of good versus bad grammar rule choices. This will enable the design generation algorithm to prefer one rule over another. For example the algorithm may decide to use a cam instead of a gear to address “change rotational mechanical energy” function in a certain design context, or decide not to add a rotational coupler if there is a bearing already existing in design, etc.

10.4.3 Design Evaluation

One of the most direct next steps includes exploring ways to evaluate the dynamics of the generated design configurations. This more rigorous evaluation of design concepts requires quantification of component geometry. This would allow the analysis of potential system behavior by means of a dynamic simulation. Since quantitative knowledge is limited during conceptual design, a qualitative simulation technique may be employed to estimate potential future states of a design configuration.

The problem of evaluating automatically generated design alternatives suggests another avenue for further study. As previously mentioned in Chapter 8, the current evaluation method is limited to a single design objective or criterion. Thus, a natural improvement area is to extent the DPM method to account for multiple objectives.

10.4.4 Level of Automation

One interpretation of increasing the level of automation during conceptual design is to create solutions that are as close to their final embodied states as possible. This can be achieved, for example, by arranging the components of a CFG in a final 3-D layout. This research is not aimed at addressing challenges involving decisions about parametric details that govern the shape, or geometry of a component. Nevertheless, bridging the gap between “topology design” and “shape design” is a challenging, open area of research. There is a great possibility for example in integrating shape grammar-based approaches with that of the CFG grammar to be able to include representations of shape in the computational approach. Such methods hold the possibility of performing individual component-based, or module-based detailed geometry design starting from an automatically generated CFG.

Additionally, one opportunity to increase the level of automation for conceptual design is to shift the focus of the implementation from technical design to industrial

design. The current conceptual design grammar is build purely upon technical design knowledge. However, product design includes other types of design knowledge. Thus, a design automation approach involving aesthetics and human factors is an interesting future research.

10.5 CONCLUDING REMARKS

The computational theory developed here bridges an important gap between computational design tools and conceptual design methods. The resulting tool assists designers with conceptual design tasks and presents an example of how design tools can be advanced to the next level of automation in order to support design creation and invention.

Appendix A: The Electromechanical Component Taxonomy

*developed in collaboration with researchers at the University of Missouri at Rolla.

Primary Component Classification	Secondary Component Classification	Component Term	Component Subset	Synonyms	Definition
Branchers	Separaters	Divider		*diaphragm, partition, panel, wall, barrier*	A device that divides a material into smaller separate areas.
		Abrasive		*polisher*	A device or material that uses texture on a surface to remove any portion of a firm (non-fluid) material.
		Blade		*cutting edge, knife, razor, scraper*	A device or material consisting of a broad flat or concave edge used to separate a firm (non-fluid) material.
		Vibrator		*sonic bath*	A device that uses frequency oscillations to separate or dislodge a firm (non-fluid) material.
		Centrifuge			A device that uses centrifugal force via a rapidly rotating container to separate fluid materials.
		Material Filter		*clarifier, separator*	A device or material consisting of a pattern of holes, slits, or pores used to separate constituents of a fluid mixture.
			Permeable Membrane		A material filter that uses a fine, porous, flexible material to separate particles from the surrounding mixture.
			Rake	*comb*	A material filter that uses a series of parallel slits or tines to separate particles from the surrounding mixture.
			Screen	*grate*	A material filter that uses a mesh structure to separate particles from the surrounding mixture.
	Distributors	Brush			A device that uses bristles to distribute a fluid material over a surface.
		Diverger			A device or structure that distributes a flow of material into multiple directions by way of its geometry.
		Nozzle			A device at the end of a pipe, hose, or tube used to distribute a continuous flow of fluid material.
		Electric Distributor			A device used to systematically allocate electrical energy along multiple paths.
	Importers/Exporters	Housing		*main body, shell, holder, casing, sheath, wrapping, enclosure*	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.
		Electric Cord		*plug and cord, power cord*	A device used to bring electrical energy into a system from an external receptacle.
	Transferors	Carousel			A device used to move material in a continuous circular path.
		Conveyor			A device used to move material in a linear path.
		Electric Conductor		*lead*	A device used to transmit electrical energy from one component to another.
			Electric Wire		An electric conductor in the form of a thin, flexible thread or rod.
			Electric Plate		An electric conductor in the form of a thin, flat sheet or strip.
		Electric Socket			A device in the form of a receptacle that transmits electrical energy via a detachable connection with an electric plug.
		Electric Plug			A device in the form of a plug that transmits electrical energy via a detachable connection with an electric socket.
		Projectile			A device that transmits mechanical energy by being thrown or propelled through the air.
		Belt		*strap, girdle, band, restraint*	A device shaped as an endless loop of flexible material between two rotating shafts or pulleys used to transmit mechanical energy.
		Clutch			A device used to transmit rotational energy between two shafts that may be (dis)engaged smoothly.

Primary Component Classification	Secondary Component Classification	Component Term	Component Subset	Synonyms	Definition
Channelers		Extension			A device that transmits mechanical energy between two elements of any jointed apparatus as they are drawn away from each other.
		Rotational Coupler		*union, compression coupling, clamping coupling*	A device used to connect coaxial shafts for power transmission from one to the other.
		Shaft		*driveshaft, output shaft, input shaft, jack shaft, half shaft*	A device in the form of a cylindrical bar used to support rotating pieces or to transmit power or motion by rotation.
		Heat Exchanger		*intercooler, platen, radiator*	A device used to transmit heat from one medium to another.
		Thermal Conductor			A device used to transmit thermal energy from one component to another.
			Thermal Wire		A thermal conductor in the form of a thin, flexible thread or rod.
			Thermal Plate		A thermal conductor in the form of a thin, flat sheet or strip.
		EM Transmitter		*transmitter*	A device that transmits electromagnetic (EM) signals (such as infrared or RF) over a non-wired medium.
	Guiders	Hinge		*pivot, axis, pin, hold down, jam, post, peg, dowel*	A device that allows rigidly connected materials to rotate relative to each other about an axis, such as the revolution of a lid, valve, gate or door, etc.
		Tube		*pipe, cylinder, conduit, channel, duct, nipple, sleeve*	A device in the form of a hollow body, usually cylindrical and long in proportion to its diameter, used to direct fluid material along a path.
		Diode			A semiconductor device which allows current to flow in only one direction.
		Bearing		*journal bearing, thrust bearing*	A device in the form of a ball or arrangement of balls that is placed between moving parts to allow them to move easily relative to each other along a path.
		Link		*connection, pawl, rod, strut, brace, cross piece, girder*	A device connecting two or more components that transmits motive power from one part to another along a specific path.
		Sled		*shoe, runner, skid*	A device either under or within a machine used to facilitate the sliding of components relative to each another along a path.
Connectors	Couplers	Clamp		*gripper, chuck, hose clamp*	A device used to hold two or more components together that is readily (dis)engageable without the use of an external tool.
		Fastener			A device used to hold two or more components together indefinitely with great effort or an external tool required to separate the joined components.
			Glue	*tape, adhesive*	A fastener in the form of an adhesive substance.
			Key	*half-moon key, cotter key, shear 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	*wing nut, lug nut, female screw*	A fastener in the form of a threaded pin that screws into a usually square or hexagonal material through a threaded hole.
			Retaining Clip	*band, hoop, loop*	A fastener in the form of a brace, band, or clasp.
			Rivet	*pop-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	*jackscrew, power screw, drive screw, lead screw, set screw, machine*	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.
	Mixers	Agitator		*stirrer, mover*	A device used to maintain fluidity and plasticity, and to prevent segregation of liquids and solids in liquids, such as concrete and mortar.
		Door		*gate, flap, access panel, entrance*	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.

Primary Component Classification	Secondary Component Classification	Component Term	Component Subset	Synonyms	Definition
Magnitude Controllers	Actuaters	Electric Switch		*knob, button, toggle*	A device for making or breaking the flow of electrical energy in an electrical circuit.
		Latch Release		*catch, pawl, lock*	A device that is designed to hold or free a mechanism as required.
	Regulaters	Valve		*louver, regulator, tap, flap valve, rotary 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.
		Potentiometer		*pot*	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.
	Changers	Mold		*form*	A hollow device used to give shape to a molten or hot fluid when it cools and hardens.
		Punch			A device used to make holes, impress a design, or stamp a die into a firm material.
		Stuffing		*padding, wadding, filling*	A device used to fill up hollows and to fill out or expand the outlines of the body.
		Choke		*throttle*	A device in the form of a restriction in a pipe that reduces the flow of a fluid material.
		Electric Resistor			A device that alters the flow of electrical energy by resisting the passage of electrical current.
		Mechanical Transformers			A device that alters the flow of mechanical energy during the process of transmitting force and motion between rotating or translating components.
			Gear	*cog wheel, rack, pinion, ring, sun, planet*	A mechanical transformer in the form of a disc or plate that transmits mechanical energy to another device by means of teeth.
			Pulley	*step pulley*	A mechanical transformer in the form of a wheel or drum fixed on a shaft and turned by a belt, chain, or strap.
			Sprocket		A mechanical transformer in the form of a toothed wheel that engages a power chain.
		Inclined Plane		*wedge*	A device in the form of a surface sloped at an angle to a reference surface, which provides a mechanical advantage for raising loads.
		Lever		*bar, peddle, rocker arm, lever arm*	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.
		Needle		*spine, stylus*	A device in the form of a slender, usually pointed, rod used to amplify a mechanical rotation on a dial or other measuring instrument.
		Lens		*convex lens, concave lens, prism*	A device in the form of a translucent substance used to alter the path of optical energy transmitted through it.
		Capacitor			A device used to alter a signal by storing an electrical charge.
		Inductor			A device used to alter a signal by storing energy as a magnetic field.
		Signal Filter			A device that alters the frequency spectrum of signals passing through it.
		Cap		*stopper, plug, bung*	A device in the form of a firm material secured to and used to prevent the flow of material into a hole or aperture.
		Cover		*top, lid, hood, shield, shroud, guard*	A device that overspreads an object, which is used to hide, defend, or shelter a material.
		Seal		*gasket, o-ring*	A device used prevent the flow of a fluid material, especially at a place where two surfaces meet.

Primary Component Classification	Secondary Component Classification	Component Term	Component Subset	Synonyms	Definition
	Stoppers	Acoustic Insulator		*silencer*	A device used to prevent the passage of sound, or vibration.
		Electric Insulator		*insulation*	A device used to prevent the passage of electrical energy.
		Fuse		*circuit breaker*	A device that breaks the flow of electrical energy in an electrical circuit in response to an excessive current.
		Cushion		*bumper*	A device in the form of a soft pad or bumper used to prevent the transmission of mechanical energy from jarring, friction, or pressure.
		Friction Enhancer		*brake pad*	A device in the form of a material used to reduce heat and increase friction.
		Stop		*snubber, travel limiter*	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.
Converters	output gas material	Catalytic Converter			A device used to chemically transform a harmful gas material into one or more inert forms.
		Evaporator			A device used to transform a liquid material into a gas material.
	output liquid material	Condenser			A device used to transform a gas material into a liquid material.
	output acoustic energy	Speaker		*woofer, tweeter, loudspeaker*	A device used to transform an electrical signal into acoustic energy.
	output electrical energy	Generator			A device used to transform mechanical energy into electrical energy.
	output electromagnetic energy	Light Source			A device used to transform electrical energy into the spectrum of electromagnetic energy visible to humans.
	output hydraulic energy	Hydraulic Pump			A device used to transform mechanical energy into hydraulic 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.
	output magnetic energy	Electromagnet			A device used to transform electrical energy into magnetic energy.
	output mechanical energy	IC Motor			A device used to transform chemical energy in the form of liquid fuel into mechanical energy.
		Electric Motor			A device used to transform electrical energy into mechanical energy.
		Hydraulic Piston			A device in the form of a cylinder tightly fitted inside a tube used to transform hydraulic energy into translational energy.
		Armature			A device used to transform magnetic energy into rotational 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.
		Wheel			A device in the form of a disc or circle used to transform translational energy applied at the hub into rotational energy.
		Airfoil			A device with curved surfaces used to transform pneumatic energy into translational energy.
		Pneumatic Piston			A device in the form of a cylinder tightly fitted inside a tube used to transform pneumatic energy into translational energy.

Primary Component Classification	Secondary Component Classification	Component Term	Component Subset	Synonyms	Definition
	output pneumatic 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.
		Pneumatic Pump			A device used to transform mechanical energy into pneumatic energy by altering the pressures within a system.
	output thermal energy	Burner			A device used to transform chemical energy into thermal energy.
		Heating Element			A device used to transform electrical energy into thermal energy.
	output control signal	Knob		*dial, button*	A device used to transform human energy into a control signal.
Provisioners	Material Suppliers	Reservoir		*cup, vessel, bucket, bottle*	A device in the form of an open tank used to accumulate and dispense a material.
		Container		*box, receptacle, holder*	A device in the form of a closed canister used to accumulate and dispense a material.
		Bladder		*balloon*	A device in the form of a hollow, expandable sac or membrane with a narrow opening used to accumulate and dispense a material.
		Pressure Vessel		*air tank, gas tank*	A device in the form of a sealed tank used to accumulate and dispense a pressurized fluid material.
	Energy Suppliers	Battery		*cell*	A device used to accumulate and dispense electrical energy by means of a chemical reaction.
		Magnet			A device used to accumulate and dispense magnetic energy.
		Flywheel		*inertia wheel, momentum wheel*	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.
Signalers	Sensors	Level Gauge			A device in the form of an external plate or face on which the amount of a fluid material is determined.
		Voltmeter			A device used to determine the voltage across a portion of an electric circuit.
		Ammeter			A device used to determine the current through an electric circuit.
		Pressure Gauge			A device used to determine the pressure from hydraulic or pneumatic energy in a system.
		Displacement Gauge			A device used to determine translational or rotational distance in a system.
		Speed Gauge			A device used to determine velocity in a system.
		EM Sensor			A device used to detect an electromagnetic signal.
	Indicators	Visual Indicator			A device used to visibly indicate a signal.
			Analog Display		A visual indicator in the form of a continuously variable dial or gauge.
			Digital Display		A visual indicator in the form of a dicrete readout or gauge.
			Flag		A visual indicator in the form of a physical banner or marker.
			Indicator Light		A visual indicator in the form of a single bulb.

Primary Component Classification	Secondary Component Classification	Component Term	Component Subset	Synonyms	Definition
		Auditory indicator			A device used to acoustically indicate a signal.
			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.
			Recording		An auditory indicator in the form of stored acoustic information that is replayed.
	Processors	Circuit Board			A device in the form of a printed circuit used to perform systematic operations on a signal.
Supporters	Stabilizers	Insert		*grommet, eyelet, bushing*	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		*stand, buttress, scaffold, brace, reinforcement, pillar, column,*	A device that holds up or sustains the weight of a body.
	Securers	Bracket		*cantilever, console, corbel, strut*	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.
	Positioners	Washer		*disk, rim, ring*	A device in the form of a disk or ring used to provide spacing between components located on a axle or shaft.
		Handle		*hand hold*	A device used to place a human hand in an appropriate configuration for grasping or interacting.

Bibliography

Agarwal, M., and J. Cagan, 1998, "A Blend of Different Tastes: The Language of Coffee Makers", Environment and Planning B: Planning and Design, Vol. 25, No. 2, pp. 205-226.

Altshuller, G., 1984, Creativity as an Exact Science, Gordon and Branch Publishers, Luxembourg.

Antonsson, E. K., Cagan, J., 2001, Formal Engineering Design Synthesis, Cambridge University Press.

Bhatta, S., Goel, A., and Prabhakar, S., 1994, "Innovation in Analogical Design: A Model-Based Approach," Proceedings of the AI in Design, Kluwer Academic, Dordrecht, The Netherlands, pp. 57–74.

Bohm, M. and Stone, R., 2004, "Product Design Support: Exploring a Design Repository System", Proceedings of IMECE'04, IMECE2004-61746, Anaheim, CA.

Bohm, M., and Stone, R., 2004, "Representing Functionality to Support Reuse: Conceptual and Supporting Functions", Proceedings of DETC'04, DETC2004-57693, Salt Lake City, UT.

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

Bracewell, R. H., and Sharpe, J. E. E., 1996, "Functional Description Used in Computer Support for Qualitative Scheme Generation- 'Schemebuilder'", Artificial Intelligence for Engineering Design, Analysis and Manufacturing, Vol. 10:4, pp. 333-345.

Brown, K.N., and J. Cagan, 1997, "Optimized Process Planning by Generative Simulated Annealing", Artificial Intelligence in Engineering Design, Analysis and Manufacturing, Vol. 11, pp.219-235.

Bryant, C., Stone, R., McAdams, D., Kurtoglu, T., Campbell, M., 2005, "A Computational Technique for Concept Generation", ASME 2005 International Design Engineering Technical Conference, Long Beach, CA.

Bryant, C., Stone, R., McAdams, D., Kurtoglu, T., Campbell, M., 2005, "Concept Generation from the Functional Basis of Design" Proceedings of International Conference on Engineering Design, ICED'05, Melbourne, Australia.

Cagan, J., 2001, "Engineering Shape Grammars," Formal Engineering Design Synthesis, Antonsson, E. K., and J. Cagan, eds., Cambridge University Press.

Cagan, J., Campbell, M., Finger, S., Tomiyama, T., 2005, "A Framework for Computational Design Synthesis", Journal of Computing and Information Science in Engineering, 5:171-181.

Cagdas G., 1996, "A Shape Grammar: The Language of Traditional Turkish Houses," Environment and Planning B, Planning and Design, 23:4

Campbell, M., Cagan J., and Kotovsky K., 2000, "Agent-based Synthesis of Electro-Mechanical Design Configurations", Journal of Mechanical Design, Vol. 122:1, pp. 61-69.

Campbell, M., Cagan, J., Kotovsky, K., 1999, "A-Design: An Agent-Based Approach to Conceptual Design in a Dynamic Environment," Research in Engineering Design, Vol. 11:3, pp. 172-192.

Campbell, M. I., 2007. "A Graph Grammar Methodology for Creative Systems", *Artificial Intelligence*, in review.

Campbell, M. I., 2006. The official GraphSynth Site, *http://www.graphsynth.com,* University of Texas at Austin

Chakrabarti, A. and Bligh, T., 1996, "An Approach to Functional Synthesis of Mechanical Design Concepts: Theory, Applications and Emerging Research Issues," Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 10, pp.313-331.

Clark-Carter, D., 1997, Doing Quantitative Psychological Research: From Design to Report, Psychology Press, Hove.

Finger, S. and Rinderle, J. R., 1989, "A Transformational Approach to Mechanical Design Using a Bond Graph Grammar," Proceedings of the 1st ASME Design Theory and Methodology Conference, Montreal.

Fu, Z., De Pennington, A., and Saia, A., 1993, "A Graph Grammar Approach to Feature Representation and Transformation", International Journal of Computer Integrated Manufacturing, Vol. 6:102, pp. 137-151.

Gliner, J., and Morgan, G. A., 2000, Research Methods in Applied Settings: An Integrated Approach to Design and Analysis, Lawrence Erlbaum Associates, New Jersey & London.

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

Hauser, J.R., D Clausing, 1998, The House of Quality, Harvard Business School, Pub. Division.

Hazelrigg, G., 1996, Systems Engineering: An Approach to Information-Based Design. Prentice Hall, Upper Saddle River, NJ.

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

Horva´th, I., Dorozsmai, K., Thernesz, V., 1994, ''A Feature-Object-Based Practical Methodology for Integration of Conceptual and Morphological Design,'' Lancaster International Workshop on Engineering Design CACD '94, Lancaster University, pp. 131–149.

Horva´th, I., Vergeest, J. S. M., and Kuczogi, G., 1998, ''Development and Application of Design Concept Ontologies for Contextual Conceptualization", ASME Design Engineering Technical Conferences, DETC98/CIE-5701, ASME, Atlanta, GA.

Hsu W, Irene M. Y. Woon, 1998, "Current research in the conceptual design of mechanical products", *Computer-Aided Design* , Vol:30:5, pp. 377-389

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

Hubka, V., Andreasen, M.M., and Eder, W.E., 1998, Practical Studies in Systematic Design, Butterworth, London.

Hundal, M., 1990, "A Systematic Method for Developing Function Structures, Solutions and Concept Variants," Mechanism and Machine Theory, 25:3, pp.243-256.

Iwasaki, Y., Fikes, R., Vescovi, M., and Chandrasekaran, B., 1993, "How things are intended to work: Capturing functional knowledge in device design". In Proceedings of the 13th International Join Conference of Artificial Intelligence, San Mateo, CA: Morgan Kaufmann, pp. 1516-1522.

Karnopp, D., and Rosenberg, R., 1975, System Dynamics: A United Approach, Wiley, New York.

Keeney, R. L., and Raiffa, H., 1976, Decisions with Multiple Objectives: Preferences and Value Tradeoffs, John Wiley & Sons, New York.

Kirschman, C. and Fadel, G., 1998, "Classifying Functions for Mechanical Design", Journal of Mechanical Design, Transactions of the ASME, 120:3, pp.475-482.

Kitamura, Y., and Mizoguchi, R., 1998, "Functional Ontology for Functional Understanding," Twelfth International Workshop on Qualitative Reasoning (QR-98), AAAI Press, Cape Cod, Massachusetts, pp. 77-87.

Kitamura, Y. and Mizoguchi, R., 1999, "Metafunctions of Artifacts," Proceedings of the Thirteenth International Workshop on Qualitative Reasoning (QR-99), Loch Awe, Scotland, pp. 136-145.

Kitamura, Y., and Mizoguchi, R., 2003, ''Ontology-Based Description of Functional Design Knowledge and its Use in a Functional Way Server,'' Expert Sys. Applic., 24, pp. 153–166.

Koch, P., Peplinski, J., Allen, J. and Mistree, F., 1994, "A Method for Design Using Available Assets: Identifying a Feasible System Configuration," Behavioral Science, 30, pp. 229-250.

Koning H., Eizenberg J., 1981, "The Language of the Prairie: Frank Lloyd Wright's Prairie Houses" Environment and Planning B: Planning and Design, 8, pp.295-323.

Kota, S., and Chiou, S.-J.,1992,"Conceptual Design of Mechanisms Based on Computational Synthesis and Simulation of Kinematic Building Blocks," Research in Engineering Design, Vol. 4, pp. 75-87.

Krishnamurti, R., 1992, "The Maximal Representation of a Shape", Environment and Planning B: Planning and Design, 19, pp. 267-288.

Krishnamurti, R., and Stouffs, R., 1993, "Spatial Grammars: Motivation, Comparison and New Results", CAAD Futures 1993, pp 57-73.

Kurfman, M., Rajan, J., Stone, R. and Wood, K., 2001 "Functional Modeling Experimental Studies," Proceedings of DETC2001, DETC2001/DTM-21709, Pittsburgh, PA..

Kurfman, M., Rajan, J., Stone, R. and Wood, K., 2003, "Experimental Studies Assessing the Repeatability of a Functional Modeling Derivation Method," Journal of Mechanical Design, 125:4, pp. 682-693.

Kurtoglu, T., Campbell, M., Bryant, C., Stone, R., McAdams, D., 2005, “Deriving a Component Basis for Computational Functional Synthesis”, Proceedings of International Conference on Engineering Design, ICED’05, Melbourne, Australia.

Kurtoglu, T., Campbell, M., Gonzales, J., Bryant, C., Stone, R., McAdams, D., 2005, “Capturing Empirically Derived Design Knowledge for Creating Conceptual Design Configurations”, ASME 2005 International Design Engineering Technical Conference, Long Beach, CA.

Li Zhanjun, D., Anderson, K. Ramani, 2005, “Ontology-Based Design Knowledge Modeling for Product Retrieval,” Proceedings of the 2005 International Conference on Engineering Design, ICED 2005, Melbourne, Australia.

Li, X., Schmidt, L., He, W., Li, L., Qian, Y., 2001, "Transformation of an EGT Grammar: New Grammar, New Designs", Proceedings of ASME 2001 Design Engineering Technical Conferences, DETC2001/DTM-21716, Pittsburgh, PA.

Liang V., Paredis, C.J.J., 2004, “A Port Ontology for Conceptual Design of Systems”, Journal of Computing and Information Science in Engineering, Vol: 4, September 2004, pg. 206-217.

Linsey, J.S., Green, M.G., Murphy, J.T., Wood, K.L., Markman, A.B., 2005, “Collaborating to Success: An Experimental Study of Group Idea Generation Techniques”, Proceedings of DETC2005, Sept. 24-28, Long Beach, California.

Little, A., Wood, K., and McAdams, D., 1997, “Functional Analysis: A Fundamental Empirical Study for Reverse Engineering, Benchmarking and Redesign”, Proceedings of the 1997 Design Engineering Technical Conferences, 97-DETC/DTM-3879, Sacramento, CA.

Longenecker S.N., and Fitzhorn P.A., 1991, "A Shape Grammar for Non-Manifold Modeling", Research in Engineering Design, 2:3, pp. 159-170.

Malmqvist, J., Axelsson, R., and Johansson, M., 1996, “A Comparative Analysis of the Theory of Inventive Problem Solving and the Systematic Approach of Pahl and Beitz”, Proceedings of the 1996 ASME Design Engineering Technical Conferences, 96-DETC/DTM-1529, Irvine, CA.

McAdams, D. and Wood, K., 2000, “Quantitative Measures for Design By Analogy,” DETC2000/DTM-14562, Proceedings of DETC2000, Balitmore, MD.

McAdams, D., Stone, R., and Wood, K., 1999, “Functional Interdependence and Product Similarity based on Customer Needs,” Research in Engineering Design, 11:1, pp.1-19.

Messac, A., 1996, “Physical Programming: Effective Optimization for Computational Design”, AIAA Journal, Vol. 34:1, pp.149–158.

Mittal, S., Dym, C., and Morjara, M., 1985, “PRIDE: An Expert System for the Design of Paper Handling Systems”, IEEE Computer, Vol.19:7, pp. 102-114.

Mortenson, M., 1997, Geometric Modeling, John Wiley & Sons, New York.

Moss J., Cagan, J., Kotovsky K., 2004, “Learning from Experience in an Agent Based Design System”, Research in Engineering Design, Vol. 15, pp:77-92.

Murdock, J., Szykman, S. and Sriram, R., 1997, “An Information Modeling Framework to Support Design Databases and Repositories”, Proceedings of DETC‘97, DETC97/DFM-4373, Sacramento, CA.

Myers, K. L., Zumel, N.B., Gracia, P., 1999, “Automated Rationale Capture for the Detailed Design Process,”, In: 11’th Conference on Innovative Applications of AI.

Navinchandra, D., Sycara, K. P., and Narasimhan, S., 1991, “A Transformational Approach to Case-Based Synthesis”, AI EDAM, Vol. 5, pp. 31-45.

Neches, R., et al., 1991, “Enabling Technology for Knowledge Sharing,” AI Magazine, 12:3, pp. 36-56.

Osborn, A., 1957, Applied Imagination, Scribner, New York, NY.

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

Otto, K. and Wood, K., 1997, “Conceptual and Configuration Design of Products and Assemblies,” ASM Handbook, Materials Selection and Design, Vol. 20, ASM International.

Otto, K. and Wood, K., 2001, Product Design: Techniques in Reverse Engineering and New Product Development, Prentice-Hall.

Otto, K., Holtta, K., 2004, “A Multi-Criteria Framework for Screening Preliminary Product Platform Concepts”, Proceedings of DETC2004, Sept. 28-Oct. 02, Salt Lake City, Utah.

Pahl, G., and Beitz, W., 1988, Engineering Design: A Systematic Approach, Springer-Verlag.

Palmer, R. S., and Shapiro, V., 1993, “Chain Models of Physical Behavior for Engineering Analysis and Design”, Research in Engineering Design, Vol. 5, pp. 161-184.

Paredis C.J.J., Diaz-Calderon, A., Sinha, R., and Khosla, P.K., 2001, "Composable Models for Simulation-Based Design", Engineering with Computers, Vol. 17, 2001, pg. 112-128.

Paynter, H. M., 1961, Analysis and Design of Engineering Systems, MIT Press, Cambridge, MA.

Pimmler, T.U., Eppinger, S.D., 1994, “Integration analysis of product decompositions”, ASME 1994 Design Technical Conferences - Minneapolis, MN, USA.

Pinilla, J. M., Finger, S., and Prinz, F. B., 1989, “Shape Feature Description Using an Augmented Topology Graph Grammar”, Proceedings of the NSF Engineering Design Research Conference, Amherst, MA, June 11-14, pp. 285-300.

Pugh, S., 1991, Total Design: Integrated Methods for Successful Product Engineering. Addison-Wesley Publishing Company, Workingham, UK.

Qian, L., Gero, J.S., 1996, “Function-behavior-structure paths and their role in analogy-based design”, Artificial Intelligence for Eng. Design, Analysis and Manufacturing, 10, pp. 289-312.

Rajagopalan, V., Bryant, C., Johnson, J., Stone, R., McAdams, D., Kurtoglu, T., Campbell, M., 2005, “Creation of Assembly Models to Support Automated Concept Generation”, ASME 2005 International Design Engineering Technical Conference, Long Beach, CA.

Rawson, K., Stahovich, T.F., 2006, “A Method for Inferring Design Rules with Explicit Bounds of Applicability”, ASME IDETC’06, Philadelphia, PA.

Rohrbach, B., 1969, “Kreativ nach Regeln – Methode 635, eine Neue Technik zum Lösen von Problemen,” Absatzwirtschaft, 12, pp. 73-75.

Rozenberg, G., 1997, Handbook of Graph Grammars and Computing by Graph Transformation - Volume 1: Foundations, World Scientific, Singapore.

Saaty, T., 1980, The Analytic Hierarchy Process, McGraw-Hill.

Sasajima, M., Kitamura, Y., Ikeda, M. and Mizoguchi, R., 1995, “FBRL: A Function and Behavior Representation Language”, Proceedings of IJCAI’95, pp. 1830-1836.

Schmidt, L., and Cagan, J., 1995, “Recursive Annealing: A Computational Model for Machine Design”, Research in Engineering Design, 7:2, pp. 102-125.

Schmidt, L.C., 1995, An Implementation Using Grammars of an Abstraction-based Model of Mechanical Design for Design Optimization and Design Space Characterization, PhD. Thesis, Carnegie Mellon University.

Schmidt, L.C., and Cagan, J., 1998, "Optimal Configuration Design: An Integrated Approach Using Grammars", Journal of Mechanical Design, Vol. 120, pp. 2-9.

Shah, J. J., 1998, "Experimental Investigation of Progressive Idea Generation Techniques in Engineering Design", Proceedings of the DETC'98, 1998 ASME Design Engineering Technical Conferences, Atlanta, GA.

Shah, J. J., Kulkarni, S. V. and Vargas-Hernández, N., 2000, "Evaluation of Idea Generation Methods for Conceptual Design: Effectiveness Metrics and Design of Experiments", Transactions of the ASME Journal of Mechanical Design, 122, pp. 377-384.

Shah, J. J., Vargas-Hernández, N., and Smith, S. M., 2003, "Metrics for Measuring Ideation Effectiveness", Design Studies, 24, pp. 111-134.

Shea, K., J. Cagan, and S.J. Fenves, 1997, "A Shape Annealing Approach to Optimal Truss Design with Dynamic Grouping of Members", ASME Journal of Mechanical Design, Vol 119, No. 3, pp. 388-394.

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

Shooter, S., Keirouz, W., Szykman, S., and Fenves, S, "A Model for Information Flow in Design", ASME Design Engineering Technical Conference Proceedings, Baltimore, MD, 2000.

Sikand, A. and Terpenny, J., 2004, "A Web-Based Environment for Managing Product Knowledge", International Symposium on Collaborative Technologies and Systems (CTSO 4), San Diego, CA.

Simon, H. A., 1969, The Sciences of the Artificial, The MIT Press, Cambridge, MA.

Sridharan, P., and Campbell, M. I., 2004, "A Grammar For Function Structures," Proceedings of ASME 2004 International Design Engineering and Technical Conference And Computers and Information in Engineering Conferences. September 28 – October 2, Salt Lake City, UT, DETC04/DTM-57130.

Stahovich, T.F., 2000, "LearnIT: An Instance Based Approach to Learning and Using Design Strategies," ASME Journal of Mechanical Design, Vol 122:3, pp.249-256.

Stahovich, T.F., Davis, R., and Shrobe, H., 1993, “An Ontology of Mechanical Devices”, Proceedings of the 1993 AAAI National Conference on Artificial Intelligence.

Starling, A.C, and K. Shea, 2003, “A Grammatical Approach to Computational Generation of Mechanical Clock Designs”, Proceedings of ICED’03 International Conference on Engineering Design. Stockholm, Sweden.

Starling, A.C, and K. Shea, 2005, “Virtual Synthesizers for Mechanical Gear Systems,” Proceedings of ICED’05 International Conference on Engineering Design, Melbourne, Australia.

Stiny G, 1980, "Introduction to Shape and Shape Grammars", Environment and Planning B: Planning and Design, Vol. 7, pp. 343-351.

Stone, R., and Wood, K., 1999, “Development of a Functional Basis for Design”, Proceedings of DETC99, DETC99/DTM-8765, Las Vegas, NV.

Stone, R., and Wood, K., 2000, “Development of a Functional Basis for Design”, Journal of Mechanical Design, 122:4, pp.359-370.

Stone, R., Wood, K., and Crawford, R., 2000, “Using Quantitative Functional Models to Develop Product Architectures”, Design Studies, 21:3, pp.239-260.

Strawbridge, B., McAdams, D. and Stone, R., 2002, “A Computational Approach to Conceptual Design”, Proceedings of DETC2002, DETC2002/DTM-34001, Montreal, Canada.

Sturges, R.H., Kilani, M., and OShaughnessy, K., 1996, "Computational Model for Conceptual Design Based on Extended Function Logic", AI EDAM, Vol. 10, pp. 255-274.

Subramanian, D., and Cheuk-San Wang, 1995, “Kinematic Synthesis with Configuration Spaces”, Research in Engineering Design, Vol.7:3, pp.193-213.

Suh, N., 1990, The Principles of Design, Oxford University Press.

Summers, J.D., and Shah, J., 2004, “Developing Measures of Complexity for Engineering Design”, Proceedings of ASME 2004 International Design Engineering and Technical Conference And Computers and Information in Engineering Conferences, Chicago, IL.

Szykman, S., 2002, “Architecture and Implementation of a Design Repository System”, Proceedings of the 2002 Proceedings of DETC2002, DETC2002/CIE-34463, Montreal, Canada.

Szykman, S., Fenves, S., Keirouz, W. and Shooter, S., 2001, "A Foundation for Interoperability in Next-Generation Product Development Systems", Journal of Computer Aided Design, Vol. 33, pp. 545-559.

Szykman, S., Racz, J., and Sriram, R., 1999, "The Representation of Function in Computer-Based Design", Proceedings of DETC99, DETC99/DTM-8742, Las Vegas, NV.

Szykman, S., Sriram, R. and Smith, S., 1996, "Proceedings of the NIST Design Repository Workshop", Proceedings of the 1996 Gaithersburg, MD.

Terpenny, J., and Mathew, D., 2004, "Modeling Environment for Function-Based Conceptual Design", ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, 30th Design Automation Conference, Salt Lake City, Utah.

Terpenny, J., 1997, "An Integrated System for Functional Modeling and Configuration in Conceptual Design", Proceedings of the Seventh International Flexible Automation and Intelligent Manufacturing Conference, Middlesbrough, U.K., pp. 69-80.

Terpenny, J., 1998, "Toward An Integrated Framework for Concurrent Engineering Design", European Design Engineer, Vol 2:1, pp. 36-38.

Thurston, D. L., 1991, "A Formal Method for Subjective Design Evaluation with Multiple Attributes", Research in Engineering Design, Vol. 3, pp. 105-122.

Ullman, D., 1993, "A New View of Functional Modeling", International Conference on Engineering Design, ICED'93, ASME, New York.

Ullman, D., 1995, The Mechanical Design Process, McGraw-Hill, New York.

Ulrich, K. and Eppinger, S., 1995, Product Design and Development, McGraw-Hill, New York.

Ulrich, K., and Seering, W., 1989, "Synthesis of Schematic Descriptions in Mechanical Design", Research in Engineering Design, Vol. 1, pp. 3-18.

Umeda, Y., and Tomiyama, T., 1997, "Functional Reasoning in Design", IEEE Expert, March-April, pp. 42-48.

Umeda, Y., Ishii, M., Yoshioka, M., Shiomura, Y., and Tomiyama, T., 1996, "Supporting Conceptual Design Based on the Function-Behavior-State Modeler", Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 10, pp.275-288.

Uschold, M., and Gruninger, M., 1996, ‘‘Ontologies: Principles, Methods, and Applications”, The Knowledge Review, 11, pp.93–136.

Ward, A.C. and W.P. Seering, 1989, “The performance of a mechanical design compiler”, ASME, Design Engineering Vol 17, pp. 89–97.

Welch, R. V., and Dixon, J., 1994, “Guiding Conceptual Design Through Behavioral Reasoning”, Research in Engineering Design, Vol. 6, pp.169-188.

Williams, B.C., 1990, “Interaction-based Invention: Designing Novel Devices from First Principles”, AAAI-90 Proceedings, Eighth National Conference on Artificial Intelligence, Vol.1, Boston, MA, pp. 349-356.

Wood K.L, Antonsson, E.K., and Beck, J.L., 1990, “Representing Imprecision in Engineering Design: Comparing Fuzzy and Probability Calculus”, Research in Engineering Design.

Wu Z., Fernández, B.R., Campbell, M.I., 2005, “Probabilistic Strategy Based Dynamic System Design Using Bond Graph and Genetic Algorithm”, International Conference of Bond Graph Modeling, New Orleans, LA.

Xu, Q. L., Ong, S. K.,and Nee, A. Y. C., 2006 “Function-based design synthesis approach to design reuse” Research in Engineering Design, Vol 17, 27-44.

Zwicky, P., 1969, Discovery, Invention, Research through Morphological Analysis, McMillan, New York.

VITA

Tolga Kurtoglu was born on June 4, 1976 to Huseyin and Gulen Kurtoglu in Ankara, Turkey. After graduating Bornova Anadolu High School in Izmir, he attended Middle East Technical University in 1994. He graduated with a Bachelor of Science degree in mechanical engineering in 1999. He entered Carnegie Mellon University in 1999 and earned a Master of Science Degree in mechanical engineering in 2001. After working as a design engineer at Dell Corporation in Austin for 2.5 years, he entered the mechanical engineering doctoral program at University of Texas in Austin.

Permanent address: Prof. Dr. Ahmet Taner Kislali Mahallesi, Gunes Sitesi, No: 3/7 06530, Cayyolu, Ankara, TURKEY

This dissertation was typed by the author.