

# Automated Synthesis of Electromechanical Design Configurations from Empirical Analysis of Function to Form Mapping

**Tolga Kurtoglu**

Automated Design Lab  
Department of Mechanical Engineering  
University of Texas at Austin  
Austin, Texas 78712-0292  
Phone: 512 471 7347  
tolga@mail.utexas.edu

**Matthew I. Campbell**

Automated Design Lab  
Department of Mechanical Engineering  
University of Texas at Austin  
Austin, Texas 78712-0292  
Phone: 512 232 9122  
mc1@mail.utexas.edu

## Abstract

For an ideal design process, designers envision a configuration of components prior to determining dimensions or sizes of these components. Given the breadth of the component space, the design of any future artifact must be carefully planned to take advantage of the diverse set of possibilities. We conjecture that computational design tools could be developed to help designers navigate the design space in creating configurations from detailed specifications of function. In this research, a methodology is developed that extracts design knowledge from an expanding online library of engineering artifacts in the form of grammar rules. From an initial implementation of 170 rules extracted from seventeen products, we demonstrate a computational process that builds new design configurations by borrowing concepts from how common functions are solved in related designs.

**Keywords:** Concept Generation; Functional Synthesis; Automated Design; Graph Grammars; Design Reuse; Knowledge Representation.

## 1 Introduction

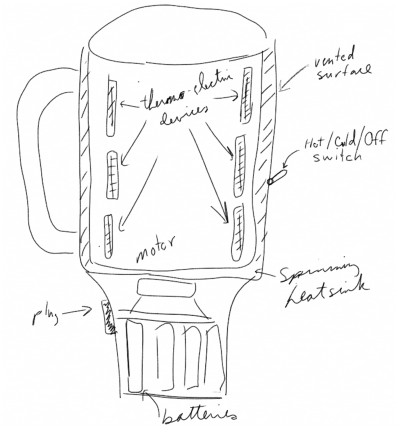
The complexity of design problems requires a structured approach to managing the concept generation process. Towards that goal, function structures developed by Pahl and Beitz (1988) provide a method that allows the designers to approach problems in smaller, easily solvable pieces. Typically, when designers use function structures, they begin by formulating the overall product function followed by decomposing it into sub-functions at lower levels of abstraction. Solutions to these sub-functions are then sought and synthesized together to arrive at a final form of a product.

For this process, a number of non-computational methods are available to help designers search solutions to sub-functions such as brainstorming, mind mapping, directed and undirected searches, and morphological analysis. However most of these are geared towards broadening and organizing solutions to sub-functions and provide little or no assistance in synthesizing potential solutions. Consider Figure 1, where a morphological matrix and a concept for a thermal mug design are shown. In the morphological matrix of the table, potential solutions to sub-functions of the design are listed in different columns. As a whole, the morphological matrix captures the potential solution space. Yet, it does not help the designer answer two important questions in moving from potential concepts to the actual configuration of the concept variant shown in the figure.

- *Which of the listed solutions should be used in a concept variant for a given sub-function?*
- *How should the selected sub-solutions of a concept variant be synthesized together into a final configuration?*

Figure 1. The morphological matrix for a “thermal mug” design and a conceptual sketch derived from it.

Morphological Matrix for Thermal Mug Design	Solution 1	Solution 2	Solution 3	Solution 4
import liquid	water tank	hose	cap	fountain machine
store liquid	water tank	bubble	cup	baby bottle
guide liquid	hose	cap	chute	levee
export liquid	water tank	hose	mouth	
stop thermal energy	styrofoam	air pocket		
import electrical energy	plug and cord	lightning rod	generator	thermo-electric device
transfer electrical energy	plug and cord	wire	circuit board	terminal block
store electrical energy	battery	capacitor	electrolitic goo	
supply electrical energy	battery	capacitor	electrolitic goo	
actuate electrical energy	switch	transistor	timer	temp sensor
transfer electrical energy	plug and cord	wire	circuit board	terminal block
convert electrical energy to thermal energy	thermo-electric device	heating plate	heat exchanger	resistor
transfer thermal energy	metal plate	ceramic disc	water	air
convert electrical energy to rotational energy	motor			
convert rotational energy to pneumatic energy	fan	blower		
import gas	fan housing	hose	straw	car AC
export gas	fan housing	hose	straw	fan blade
import human material	handle	housing	strap	
guide human material	handle	housing	strap	
export human material	handle	housing	strap	
import mechanical energy	housing	cap	handle	
distribute mechanical energy	housing	bottom cap		
export mechanical energy	bottom cover	housing		
import human energy	switch			
convert human energy to control signal	switch			



These questions summarize the crucial synthesis step of conceptual design, in which function is mapped to form. During this process, there are many instances where the designer makes conscious and unconscious decisions to select and refine concept variants. 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 designer’s work. Heuristics applied by designers in transitioning from functional requirements to conceptual design configurations can be extracted by studying existing products. We conjecture that through systematic dissection of consumer products, a methodology can be developed that extracts design knowledge employed in the creation of designs. Unlike other research that attempts to automatically synthesize a design configuration, this research leverages an expanding online repository of products from which design rules are developed to capture the knowledge of the original designer’s intent. The rules created from the repository are then initiated in a computational search process. The resulting computational design tool 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 would both increase the efficiency of the design process and the rigor of creating new solutions. Furthermore, it can remove the psychological bias that limits designers to previous solutions or to specific engineering domains.

In this paper, we present our approach to developing a formal language for generating design configurations from a functional description. Based on an empirical analysis of seventeen consumer products, 170 grammar rules are created to capture feasible mappings from product function to product form. Using the grammar-based computational tool, a design team will be able to rapidly generate multiple and feasible design configurations by leveraging expertise of past designers from variety of domains.

In the remainder of this paper, we examine state of the art in conceptual design research (Section 2) and detail some background work that led to the development of the method (Section 3). This is followed by a description of the methodology (Section 4) and a summary of our resulting rule set (Section 5). Then, we show the implementation of the rules through an illustrative example (Section 6) and we conclude with a discussion of the implications of this example and our future research goals (Section 7).

## 2 Review of Related Work

In this section, we review the state of the art in conceptual design research and areas that support automated concept generation. First, we begin with reviewing systematic approaches to conceptual design and then focus on product function representation. Secondly, we summarize various computational design methods including graph grammars.

The fuzzy front-end of the conceptual design process has seen few attempts at automation due in part, perhaps, to the evolving strategies and methodologies that exist for this phase of design. However, over the past few decades, design methods have matured and systematic approaches to conceptual design have emerged (Hubka and Eder 1984; Pahl and Beitz, 1988; Suh, 1990; Ulrich and Eppinger, 1995; Ullman 1997; Otto and Wood, 2001). These design methods provide a structured and formal approach to performing design and provide a starting point for automating the conceptual design phase. These methods are bottom-up in nature, and they all begin by formulating the overall product function and decomposing it into lower level sub-functions, solutions to which are then synthesized together to arrive at a final design. Our computational approach to concept generation also follows this function-based synthesis framework.

One of the critical requirements for function-based synthesis approach is the representation of functional knowledge. Several researchers, institutions and corporations have developed systems to capture abstract engineering design models of functionality and interface requirements. The captured models range from detailed function and behavior information to loose hierarchical artifact relationships. Among those, the NIST Design Repository Project is a framework capable of storing component information and how the elements of information are related to each other (Murdock et al, 1997; Shooter et al, 2000; Szykman, 2002). Within the NIST model there are five sections: Artifacts, Functions, Forms, Behaviors and Flows, which contain information relative to their denoted name. Similar function-based representations have been proposed by others to support search and modeling processes of conceptual design (Terpenny, 1997; Terpenny and Mathew, 2004; Sikand et al, 2004). Japanese researchers have also explored a consistent language for describing the functionality of products and relating it to product behavior (Sasajima et al., 1995; Umeda and Tomiyama, 1997; Kitamura and Mizoguchi, 1998).

Building on these efforts, the concept of a functional basis is developed by Stone and Wood (Stone and Wood, 1999) which significantly extends previous research (Little, et al., 1997; Otto and Wood, 1997). The functional basis includes a set of terms that span the space of all functions and all flows (Hirtz et al, 2002). Here, a function refers to a transformation operation from input flow to output flow. Functions are used in verb-object format. For example a motor “converts electrical energy to mechanical energy.” Three sets of function terms are defined to allow three levels of abstraction for allocating functions to a system. Tables 1 and 2 show a portion of the functional basis. Using this functional vocabulary, device function can be defined for a given system using a function structure.

Built upon this methodology, 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 to ours in principle, but uses matrix-manipulation algorithms to generate and rank conceptual solutions.

Another computational tool developed for concept generation is the previous work by the authors in the A-Design research (Campbell et al, 2000). 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. In this way, the research is similar to other dynamic simulation methodologies such as those presented in Paredis, et al (2001) and Bracewell et al (1996).

**Table 1. Flow classes and their basic categorizations.**

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

**Table 2. Function classes and their basic categorizations.**

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

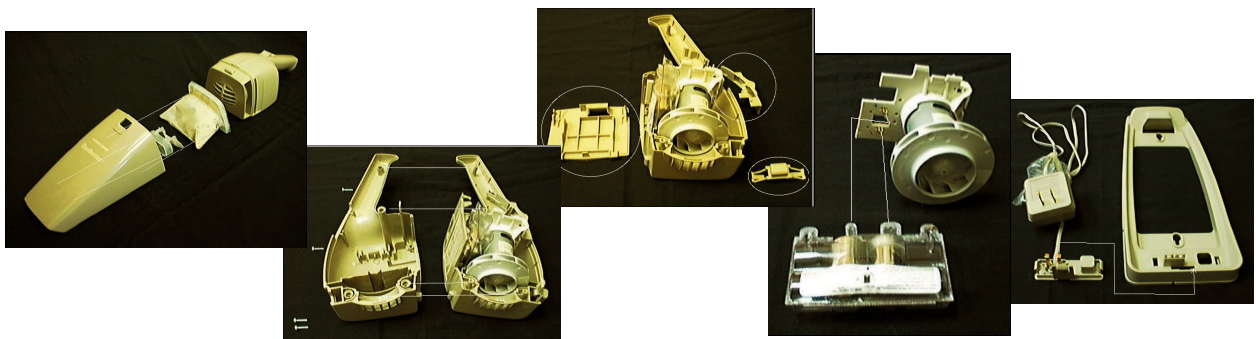
In recent years, engineering researchers have discovered that graph grammars (Rozenberg, 1997) and shape grammars (Stiny, 1980) provide a flexible and yet structured approach to engineering design methods (Cagan, 2001). This aspect makes grammars an emerging concept in design synthesis (Schmidt et al, 1995; Schmidt and Cagan, 1998; Fu, 1993; Li et al, 2001; Pinilla et al, 1989; Starling and Shea 2005). The concept of a grammar is that an experienced designer can construct a set of rules to capture his/her knowledge about a certain type of artifact. Grammar based design systems offer the option of exploring the design alternatives as well as automating the design generation process. Graph grammars are comprised of rules for manipulating nodes and arcs within a graph. These techniques create a formal language for generating and updating complex designs from a simple initial specification, or seed. The development of these rules encapsulate a set a valid operations that can occur in the development of a design. Such representations can produce a wider variety of candidates since solutions only need to have a common starting point. Built upon this characteristic, Sridharan and Campbell, (2004), 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. In this work, we start with a function structure and seek multiple configuration solutions for the various functions of the function structure. To achieve that, we combine the formal function based synthesis method (Pahl and Beitz, 1988; Ullman, 1997; Hubka and Eder 1998; Otto and Wood, 2001) with graph representations developed to facilitate the formulation of a graph grammar language for building feasible design configurations.

### 3 Research Background

Extending the previous function-based design research, we develop a design method that transforms a high-level, functional description of a non-existent product into a set of concept variants. The main hypothesis in this research is that successful designs often come from experienced designers. Under this premise, experience in the form of design knowledge is extracted from existing products and stored for reuse in a web-based repository. Three major tasks in capturing and organizing design knowledge are; studying existing artifacts, recording and storing of design knowledge in the repository, and the development of standardized vocabularies to represent function and form knowledge. The following sections explain each of these tasks.

#### 3.1 Systematic Product Teardowns

The first step in our research is the dissection of consumer products. In this step, we emphasize concrete experience with the existing product. We strive to fully understand the product in terms of functionality, components, product hierarchy, manufacturing, and assembly. Each product is completely dissected and documented. As the product is disassembled, a bill of materials (BOM) is constructed to document the components. Each component is labeled and photographed as it is removed from the product and recorded in a tabular format listing its attributes such as assembly name, part number, quantity, part description, part function(s), input-output flows, physical parameters and predicted manufacturing processes.



**Figure 2. The dissection of a Black & Decker hand held vacuum cleaner.**

In this study, seventeen consumer products are chosen to provide an empirical basis for the development of our method. These products offer simple technologies, yet the variety of the technologies used by the products cover a rich set of engineering solutions and domains. Figure 2 shows one of these products, a hand held vacuum cleaner, at various stages of teardown. The complete list of products that were analyzed is shown in Table 3.

**Table 3. The 17 products used as an empirical basis for the development of our method.**

Products Studied	
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	

### 3.2 Design Knowledge Repository

The web-based design knowledge repository<sup>1</sup>, managed at the University of Missouri-Rolla, serves as a forum to store and organize the design knowledge derived from product teardowns. The knowledge contained in the repository is steadily expanding and currently includes detailed information on approximately 70 consumer products.

The design data is recorded into the repository using a FileMaker template. Following entry and storage into FileMaker, a platform-independent data set is output as XML. By creating JSP (Java Server Pages) the XML from the database server can be viewed as HTML through a standard web browser (Bohm et al., 2004) as shown in Figure 3. Given this format, our research has leveraged the repository data to extract a design grammar language that captures the relationship between specific functions and solution principles (or components) that are used to fulfill them.

### 3.3 Design Knowledge Representation


To derive uniformity and consistency in representing design knowledge, we have adopted two standardized vocabularies. At a functional level, we use the aforementioned functional basis language developed by Hirtz et al. (2002). This standardized lexicon of terms define a comprehensive set of function and flow names at three levels of abstraction to support both high and low level functional modeling independent of the physical structure of the artifact. Using this functional vocabulary, we define device function for a given product by building a function structure.

Based on the success of defining a canonical list of function and flow names as shown in Tables 1 and 2, we sought to define a canonical list of component names (Greer, et al 2003; Kurtoglu, et al, 2005). In developing this component-naming scheme, we took a perspective that promotes function as the fundamental ontology of a component. We believe this is especially valid for the conceptual phase of design, where efficient indexing, search and retrieval of information are facilitated by function-based queries.

According to the basis, each component identified through the dissection phase is classified under a specific component name. The basis allows for well-defined groupings of components to be used in the creation of various design representations and design tools. It also eliminates component redundancies that may not be immediately evident due to variations in user-dependent component naming. By eliminating these redundancies, a larger variety of unique concept variants can be quickly generated. Through a similar dissection of products, we have converged to a set of 110 distinct component names. Table 4 shows a partial list of the component basis.

<sup>1</sup> <http://function.basiceeng.umr.edu/repository>

**System: \*Delta - Circular Saw**

<b>Artifact Name</b>	right housing	<b>Artifact Photo</b>  click on image for full size
<b>Part Family</b>	not specified	
<b>Part Number</b>	3	
<b>Sub Artifact Of</b>	housing assembly	
<b>Quantity</b>	1	
<b>Description</b>	not specified	
<b>Artifact Color</b>	not specified	
<b>Component Naming</b>	not specified	

there are no flows defined for this artifact

Supporting Functions			
grip	solid material	position	solid material
			internal

Physical Parameters		Manufacturing Process	
length	10.0	material	not specified
width	1.75	no process specified	
height	7.5		

<b>Primary Identifier</b>	no primary identifier specified	<b>Failure Mode</b>	no failure mode specified
---------------------------	---------------------------------	---------------------	---------------------------

**Component Relation Equation:**  
not specified

**Equation Metric:**

Variable	Variable Name	Variable Value	Variable Metric
no variables specified			

Figure 3. The UMR Design Repository web interface.

Table 4. A partial list from component basis.

Name	Synonyms	Definition
Acoustic Insulator	silencer	The material that provides the condition of being isolated by non-conductors to prevent the passage of sound, or vibration.
Agitator	stirrer, mover	A mechanical device used to maintain fluidity and plasticity, and to prevent segregation of liquids and solids in liquids, such as concrete and mortar.
Airfoil	wing	Each of the limbs or structures by which an animal or manmade craft is able to generate a lifting force.
Axle	stub axle, beam axle, axle shaft	A supporting member designed to carry a wheel that may be attached to it, driven by it, or freely mounted on it.
Battery		A device that produce a direct current by converting chemical energy to EE. Mostly used to store EE.
Bearing	journal bearing, thrust bearing	Any part of a machine or device that supports or carries another part that is in motion in or upon it, such as a journal bearing or thrust bearing.
Belt	strap, girdle, band, restraint, strip	An flexible band made of leather, plastic, fabric, or the like that is used to convey materials or to transmit rotary motion between shafts by running over pulleys with special grooves.
Bladder	baloon, inner tube, membrane	A device resembling any of various sacks found in most animals and made of elastic membrane.
Blade	cutting edge, knife, razor, scraper	The broad flat or concave part of a machine that contacts the material to be moved or cut.
Bracket	Cantilever, console, corbel, strut	A piece or combination of pieces, usually triangular in general shape, projecting from, or fastened to, a wall, or other surface, to support heavy bodies or to strengthen angles.
Burner		The component of a fuel-burning device, such as a furnace, boiler, or jet engine in which the fuel and air are mixed and combustion occurs.

## 4 Research Approach:

In this section we present our approach for creating new design configurations from functional requirements. We leverage the expanding online library of products and we derive design rules from it to capture the knowledge of the original designer's intent. In creating the rules, we observe the common uses of the components in the library and formulate this knowledge as design "grammar rules".

Our computational method is initiated with a completed function structure, such as the one shown in Figure 4.a. The output of the program is a design configuration(s) at a conceptual level. We represent these design configurations using a graph format, called the configuration flow graph or CFG. Like function structures, configuration flow graphs are typical graph structures represented by a set of nodes (vertices) and arcs (edges). In a CFG, nodes of the graph represent product components, whereas arcs represent flows. For flow naming, the functional basis terminology is adopted, while the components of the graph are named using the standard names of the component basis.

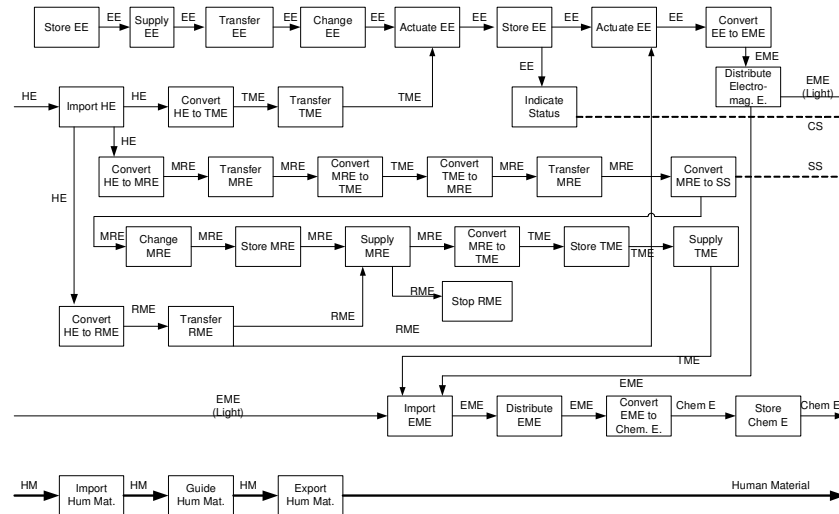


Figure 4.a. The function structure of a disposable camera.

The CFG shows individual components of a design, how the components are connected, and energy, material and signal flow interactions between them. At an abstract level, it also represents the behavior of components by modeling them as "black box" entities that transform certain input flows into certain output flows. In this way, the behavior representation is similar to port based methodologies such as those presented in (Paredis et al, 2001; Liang and Paredis, 2004). As a whole, the CFG captures the conceptual solution of a design problem. Figure 4.b. shows an example of a CFG created for a disposable camera product.

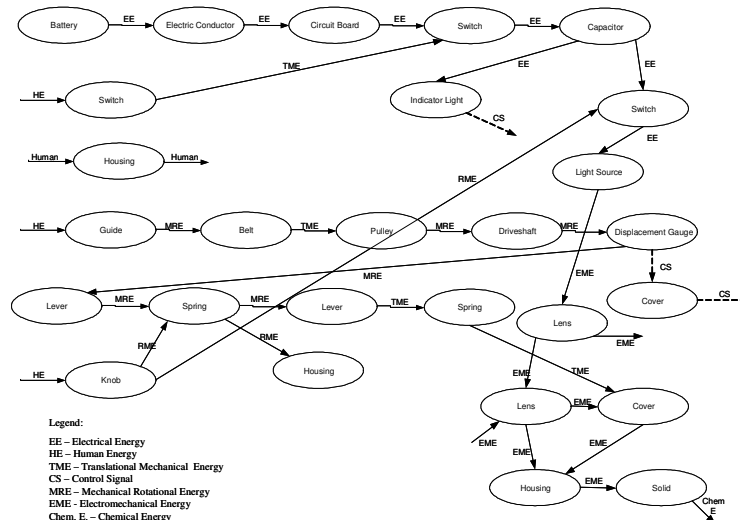


Figure 4.b. The configuration flow graph (CFG) of a disposable camera.

The diagram illustrates the transformation from a Function Structure (LHS) to a Configuration Flow Graph (RHS).

**LHS: Function Structure**

The LHS shows a sequence of functions. Function 1 receives Flow 1 and Flow 4 as inputs and outputs Flow 2. Function 2 receives Flow 2 as input and outputs Flow 3.

**RHS: Configuration Flow Graph**

The RHS shows a sequence of components. Component 1 receives Flow 1 as input and outputs Flow 2. Component 2 receives Flow 2 as input and outputs Flow 3.

A large blue arrow indicates the transformation from the LHS to the RHS.

In detail, the transformation from the function structure to CFG is part of the recognize-choose-apply cycle shown in Figure 6. Given an initial function structure,  $S$ , we first recognize all possible rules that have their LHS as a subset of  $S$ . This recognition step is followed by a choosing of one of the valid options. The options could be presented to the user who then selects one of the rules to apply or the selection could be performed automatically. In application, the CFG is updated as per the instructions provided in the RHS. This process is repeated until there are no more rules that can be applied. Note that the same right-hand side could have multiple left-hand side alternatives. These cases are represented as separate rules. This ensures enumeration of different component alternatives for a given sub-function or set of sub-functions.





## 5 Rule Set

In this section, we discuss the rule set that is developed and the methods we use to generate them. Our approach to developing these rules is based on design knowledge extracted from the aforementioned design repository. Through systematic product teardowns and data gathering in the repository, we capture an existing product's CFG and its function structure. Then we extract the relationship between these two graphs by matching components to functions. By examining these relationships we carefully construct rules that capture the ways in which components fulfill various functions.

One of challenges we faced in our study was determining the level of granularity of the function structures created for analyzed products. In the functional basis, three levels of abstraction are developed to represent functions and flows: primary, secondary, and tertiary. For formalizing our grammar definition we have adopted the secondary level. According to this, functional modeling of the products is performed using function and flow terms shown in Tables 1 and 2 only. One exception to this is that of mechanical energy, which is represented at the tertiary level, allowing us to differentiate between rotational and translational mechanical energy.

Figure 7 illustrates derivation of rules from analysis of an electric toothbrush product. As a first step, we create a function structure and the CFG of the product. Then, we capture the mapping between the two graphs. Each mapping represents a potential rule that shows how a functional requirement was transformed into a physical form solution in the actual design. Some of the rules derived from the electric toothbrush are shown at the end of Figure 7. In the first rule, the rule recognizes “convert rotational mechanical energy to translational mechanical energy” and “transfer translational mechanical energy” and their associated flows in the function structure. If the two functions and three flows are recognized, it adds the component “link” to the CFG at the appropriate location. Similarly, the last rule shown in Figure 7 recognizes the function “transfer RME” in the function structure and adds a “driveshaft” and a “rotational coupler” to the design configuration.

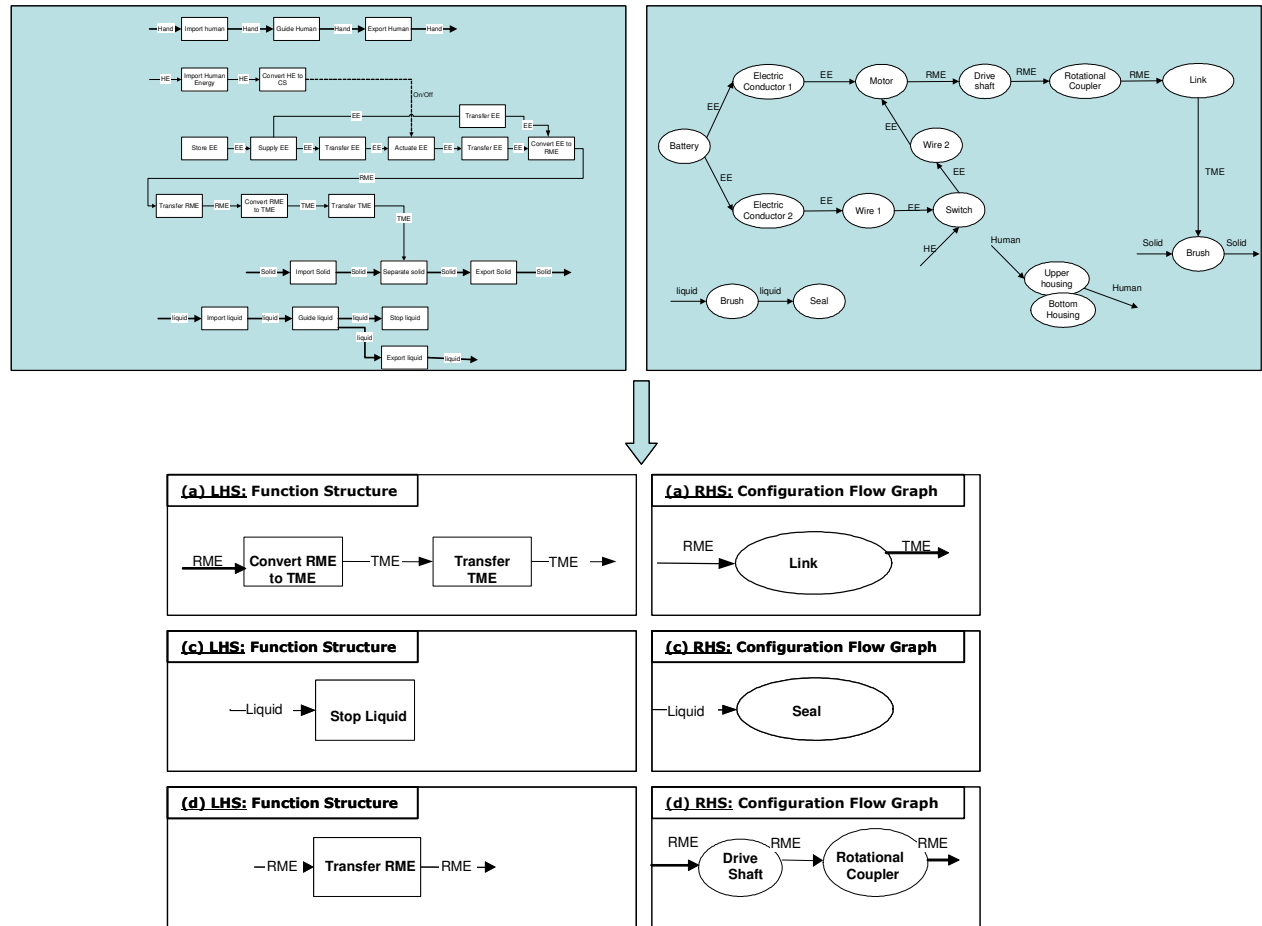


Figure 7. Illustration of rule derivation (only 3 rules) from empirical analysis of a toothbrush product.

Note that the rules in Figure 7 are not simply one-to-one matches of functions to components. The open-endedness of the grammar formulation allows us to escape 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 inserting 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.

This construction of 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 and valid rules are added to the final rule set. Furthermore, our rule definitions prevent the same rule from being applied over and over again. Figure 8 shows the graph between products examined and the rules obtained from them. The rate of newly generated rules appears to be reducing with each additional dissection. This convergence suggests that a finite set of rules may describe the function-to-form mapping in our current product domain. Our 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 domains or to other design contexts. Currently, we have 170 rules derived from 17 products. The complete list of 170 rules is shown in detail at [http://www.me.utexas.edu/~adlab/cfg\\_grammar.htm](http://www.me.utexas.edu/~adlab/cfg_grammar.htm). In the next section, we illustrate and discuss how these rules are used to create design configurations for a new design problem.

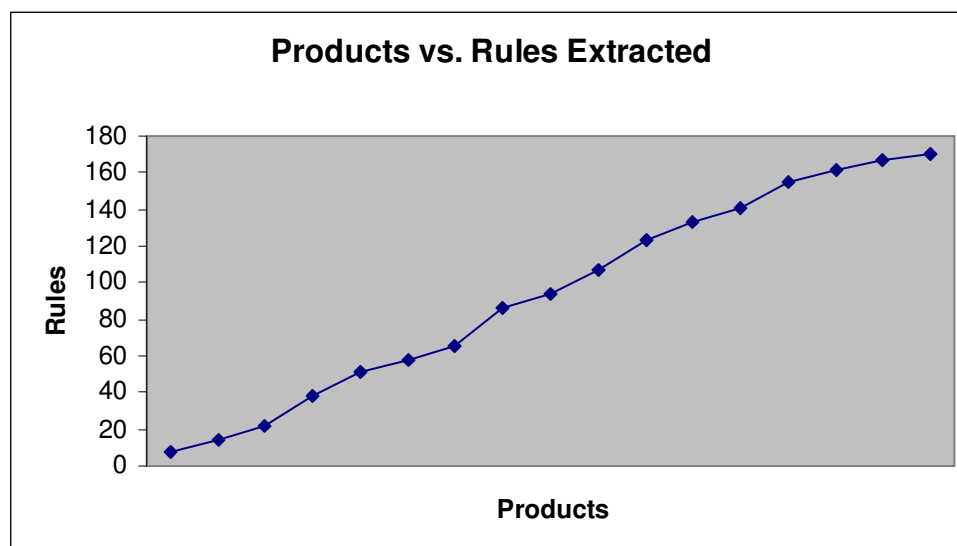


Figure 8. A graph between the number of products examined and the rules obtained from each of them.

## 6 Illustrative Example

In this section, we illustrate the implementation of our grammar for the design of a wall climber toy product. Consider the following design case:

*“A company has begun marketing a wall coating that contains ferrous micro-metal chips. This coating is “attractive” to magnetic devices and walls coated with this product “look” metallic. One potential marketing plot for the company to increase sales of its coating product is to sell a toy that would operate on the vertical space of the walls. Thus, they seek prototype toy products that use walls covered with the coating as their play space. Since there are numerous types of potential toys for this new application, this call for products is fairly open ended. Broad requirements include the ability to remain stationary while on the wall, be lightweight, have a long lasting power source and be easy to set up.”*

The function structure of the product to be designed is shown in Figure 9. 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.

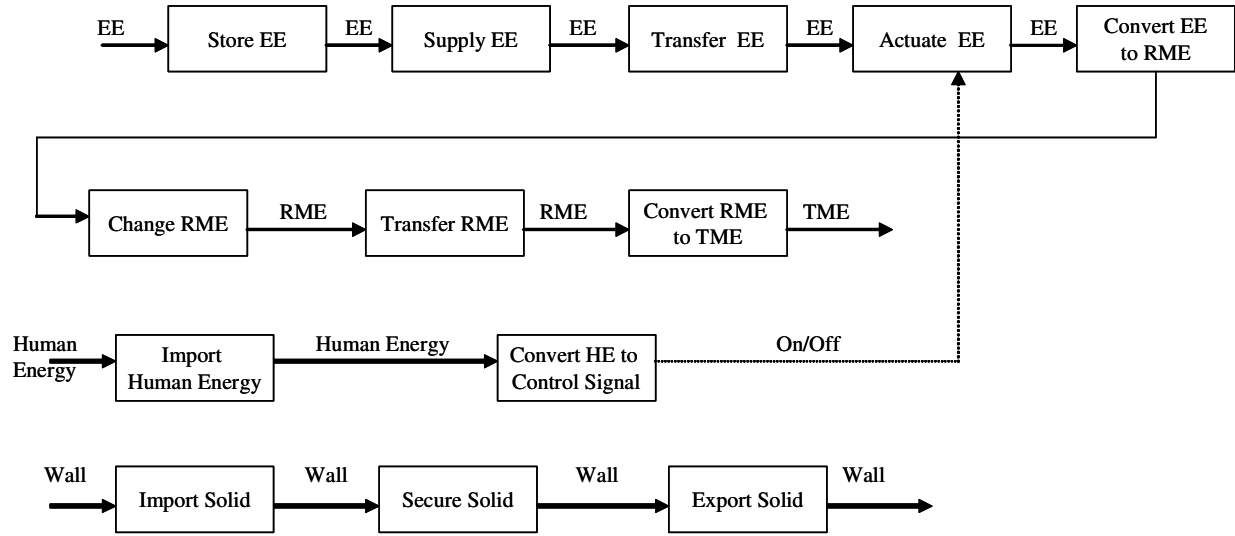
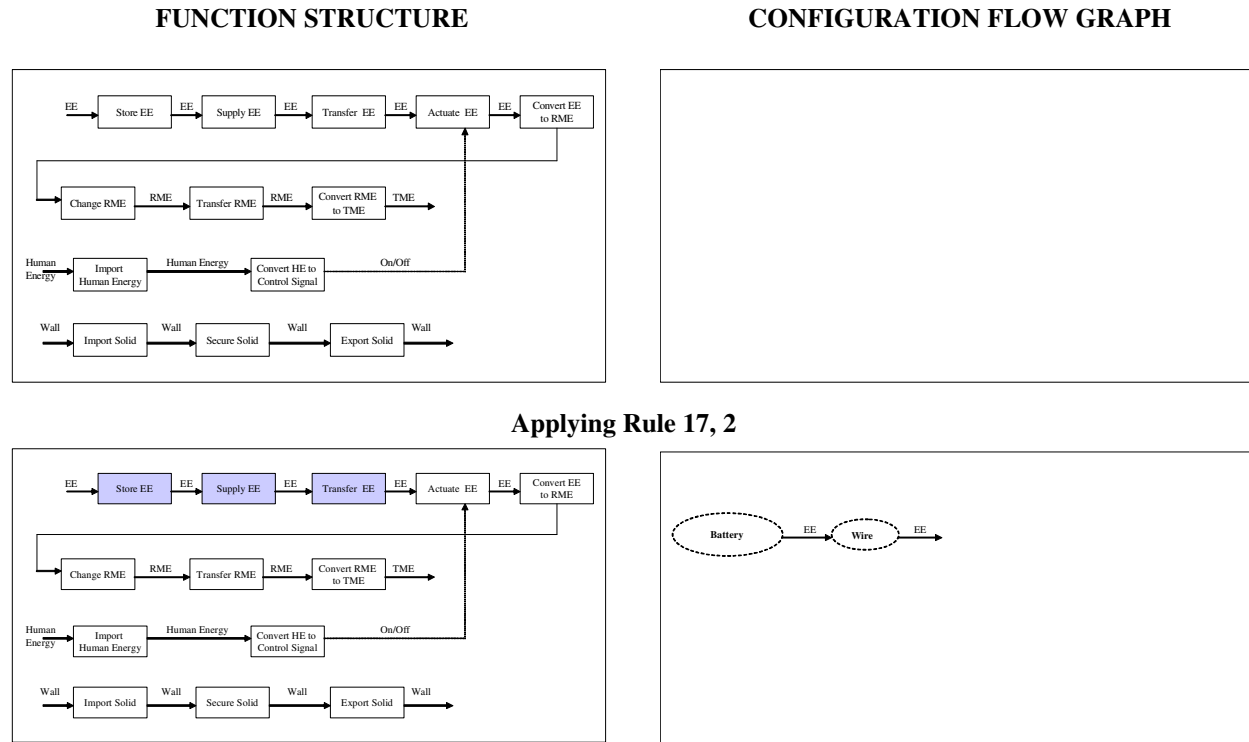
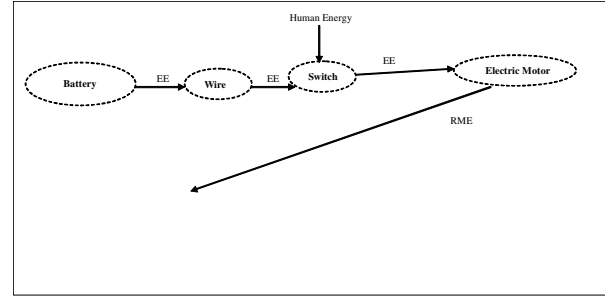
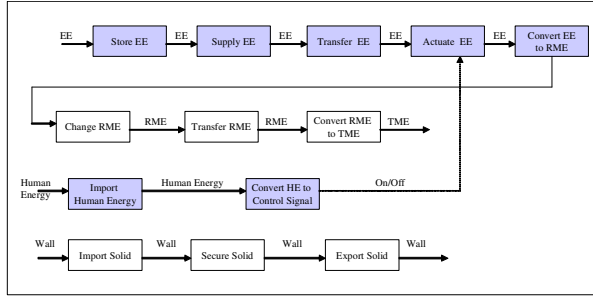


Figure 9. The function structure for the wall climber toy design problem.

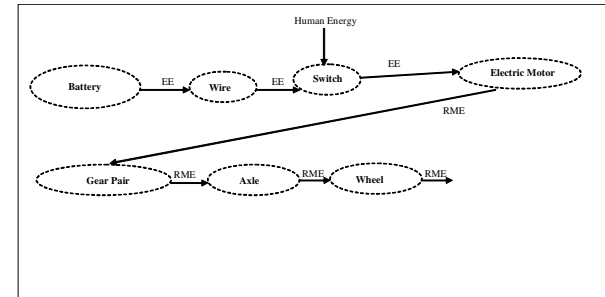
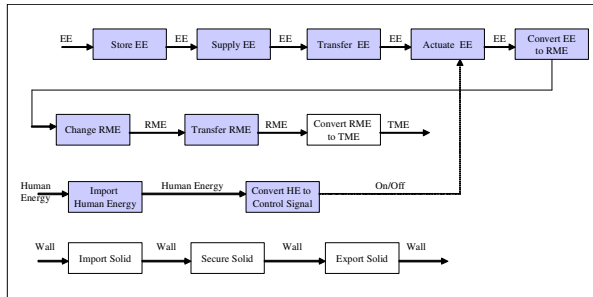
The algorithm first reads the function structure and replaces sub-functions with components after application of each new rule until no further rules can be recognized. Figure 10 shows this process. At the beginning, none of the sub-functions are mapped to components. Therefore, the CFG starts as an 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, we list the grammar rules that are applied. 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.



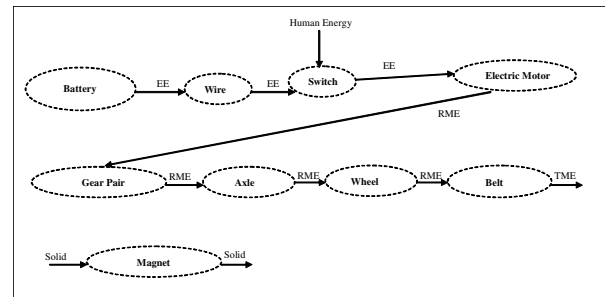
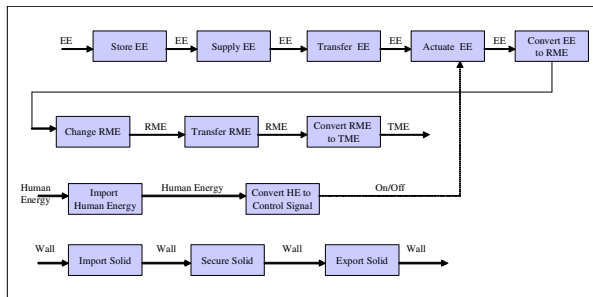
### Applying Rule 38, 14



### Applying Rule 111, 96



### Applying Rule 7, 81



**Figure 10. A pictorial representation of building the CFG a from function structure using the rule set.**

In creating this design configuration, the program successfully integrates component concepts from different products into one complete concept variant. For example, in an original sprinkler design, the hydraulic energy of the water is converted to rotational mechanical energy before it is transmitted to the wheels of the device through a gear pair and an axle. In the design generated by the program, it is suggested that the same gear-axle pair be used for rotational energy transmission. As a second example, consider the magnet that is suggested to be used in the design of the climber toy to interface with the wall. This concept is borrowed from a can opener design, where a magnet is used to hold and secure a can lid to the opener, before and after a blade cuts through it. The configuration of Figure 10 also includes concepts inherited from a single use camera (belt), a toothbrush (battery), an electric knife (wire) and a hand held vacuum cleaner (switch). Figure 11 shows two additional configurations generated by the execution of grammar rules. Overall, these three configurations include partial conceptual solutions from 11 different products.

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.

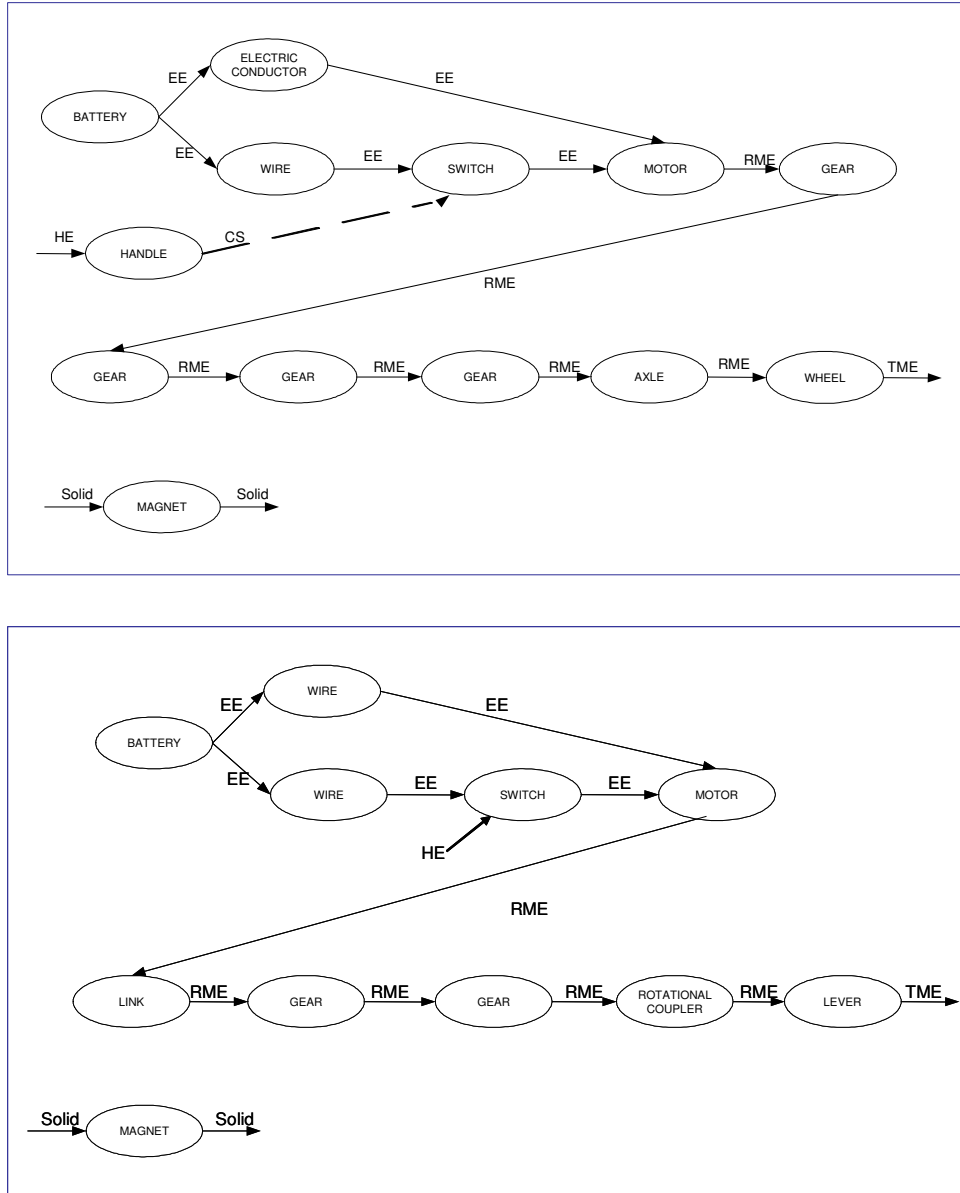


Figure 11. Additional concepts generated for the wall climber toy design.

## 6.1 Experimental Study

To further assess the effectiveness of our automated concept generator as a design aid, we have conducted an experiment with undergraduate research students. In the experiment, we asked students to generate concepts for the same wall climber toy design problem using the traditional morphological matrix method. The students first created morphological matrices and then sketched concept variants derived from them. No specific instructions were given to the students as to which concepts to select from the created morph matrices and how to configure individual solutions into complete design concepts.

The objective of the experiment was to compare the results obtained from a traditional concept generation method with that of our automated concept generator. Figure 12 shows an example morphological matrix and a design concept synthesized from it by a group of students. The highlighted sub-solutions of the morph matrix are included in the final sketched concept.

Morphological Matrix for the Wall Climber Toy				
	Solution 1	Solution 2	Solution 3	Solution 4
store electrical energy	battery	capacitor		
supply electrical energy	battery	capacitor		
transfer electrical energy	wire	metal plate	electric conductor	
actuate electrical energy	switch	transistor	circuit board	
convert electrical energy to rotational energy	motor			
change rotational energy	gear	pulleys	lever	
transfer rotational energy	driveshaft	belt	link	axle
convert rotational energy to translational energy	wheel	half-tracks	link	
import solid material	gripper	latch	magnet	housing
secure solid material	nut-bolt	guide	magnet	
export solid material	gripper	latch	magnet	housing
import human energy	joystick	knob	handle	
convert human energy to control signal	circuit board	switch	handle	

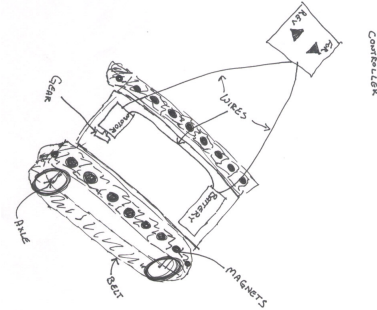


Figure 12. A morphological matrix and a concept sketch drafted from it.

For this group, the results show that, students using a traditional method, created very similar concepts to the ones generated by the execution of grammar rules. Consider the sketched concept variant of Figure 12. It has 8 components in common with the automatically generated configuration of Figure 10. The main difference is in the control and activation of the device. The students decided to use a “joystick” and a “circuit board controller” to actuate and to maneuver the toy, whereas the concept generated by the grammar includes only a simple “on/off switch” to address the same functionality. In addition to having a high percentage of shared components, the two solutions are also topologically similar. According to this, the connectivity of the components and the energy, material and signal flows through the components are nearly the same in two designs.

The results simply illustrate that a thorough formulation of the grammar rules can capture the intelligence employed by designers during the crucial synthesis step of conceptual design.

## 7 Implementation

An algorithm has been created and programmed in C# using Microsoft Visual Studio that follows the flowchart shown in Figure 13. The graphical user-interface of the program is coded using Netron, an open source resource that provides graph libraries and related software tools for programmers to develop graph and diagram based applications. Our automated concept generation software uses Netron’s standard tools to provide designers a natural visual interface.

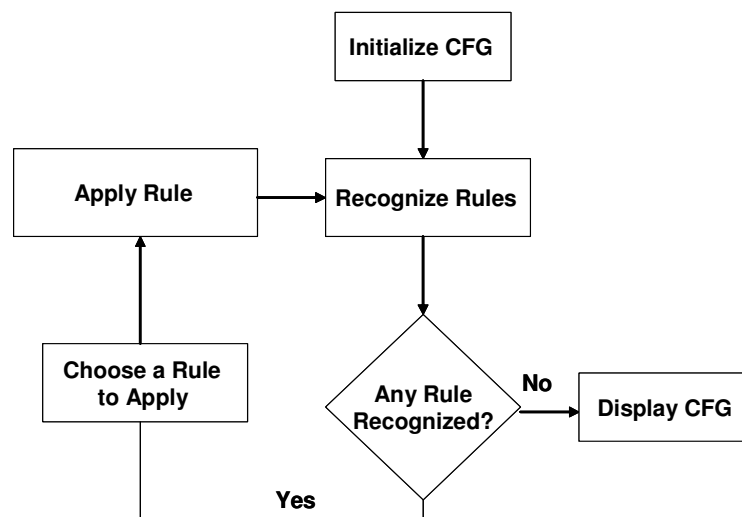
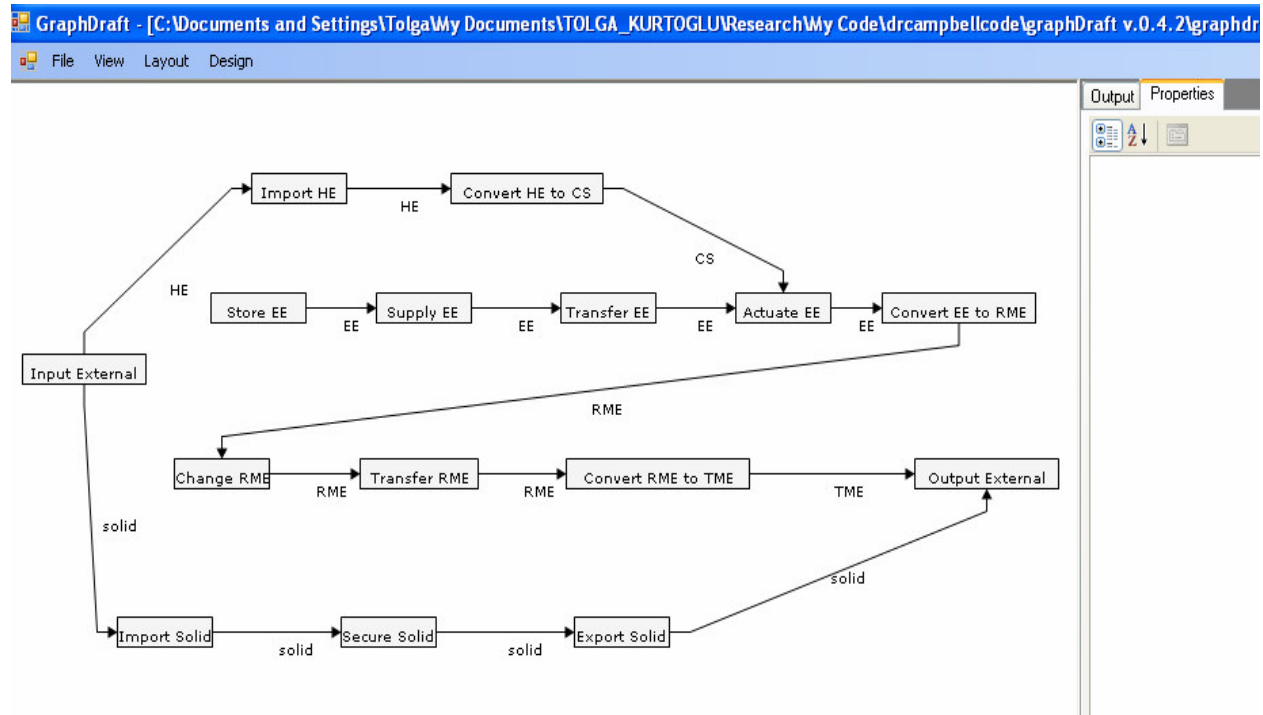


Figure 13: Detailed flowchart of rule processing.

The user interface allows the designer to quickly draft a function structure that represents the conceptual design problem. The program then converts this function structure to XML format and starts the design generation process. Figure 14 shows the user interface with the initial function structure for the wall climber toy product. 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 Netron and are rarely considered as valid connections in many graph theories and applications.



**Figure 14. Graphical user interface of the developed automated concept generation software. The designer sketches the function structure of the system that is to be designed.**

The program then executes a recognize-choose-apply loop to implement the rules. A major issue in the implementation of the rules is the recognition of where and when each rule can be applied. 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 170 rules and checking each for applicability. Any rule that satisfies recognition conditions is then listed as a “recognized” rule along with its corresponding location.

Recognition is followed by the selection of the rule to be applied. The rule that is actually applied can be a choice of either the user or an automated process. In our current implementation, the user interacts with the program via a simple dialog window, through which user’s choices for the rule(s) to be applied can be entered. This dialog window is shown in Figure 15.

The third and final step of rule processing is applying a selected rule. After the user makes a selection from the recognized rules list, the program 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 16 shows the state of the graph, after the application of “wire” and “battery” rules. In intermediate states, such as the one shown in Figure 16, 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.

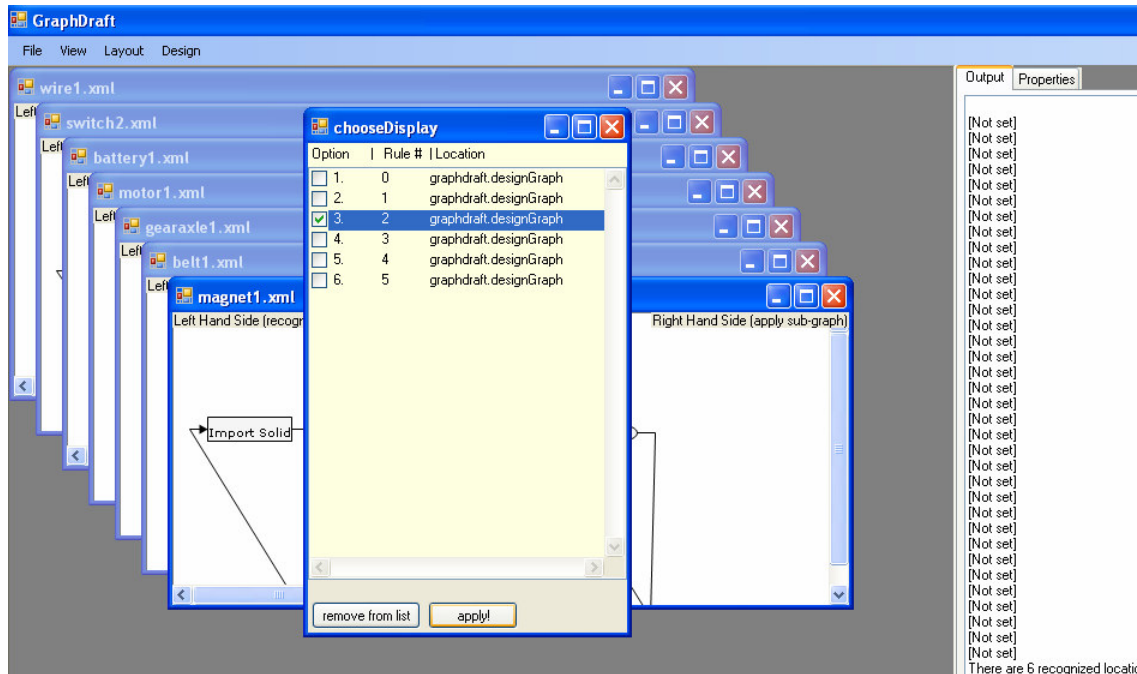


Figure 15. Dialog window for rule selection.

The described graph transformation is implemented using the “double push-out method” (Rozenberg, 1997). Double push-out is an algebraic approach to graph transformation used in various graph grammar applications. The extension of this method to engineering design problems at a generic level constitutes another research project conducted by our research group.

The program manages the rules and their applications until no further rules can be applied, thus terminating when a complete design configuration has been built for the given functional description. At the end of the process, the program displays the resulting CFG on the screen. This CFG for the wall climber toy is shown in Figure 17.

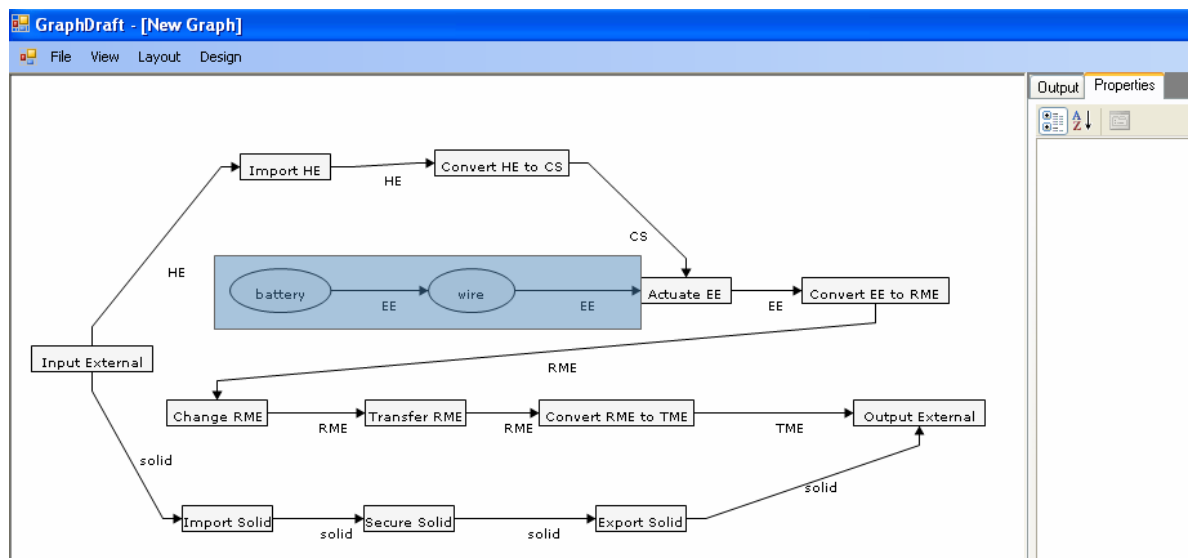


Figure 16. A screenshot of the user interface at a partially completed design state.



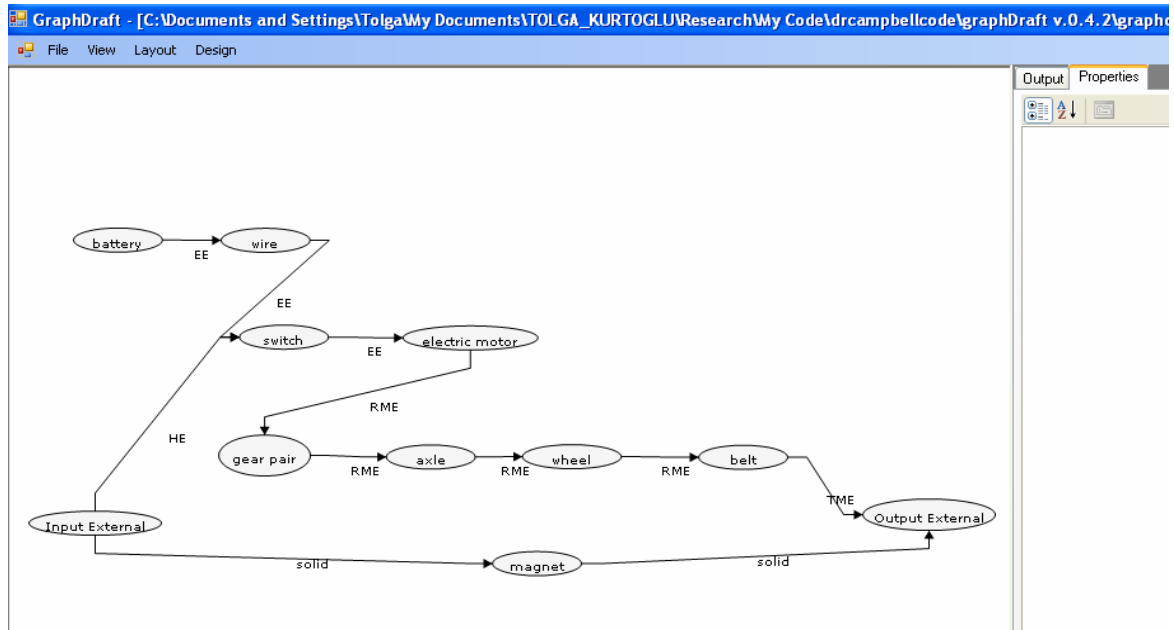


Figure 17. A screenshot of the user interface after completion of design.

## 8 Conclusions and Future Work

In this research, we are proposing a computational methodology for creating conceptual design configurations. Our method is based on leveraging existing design knowledge by integrating design concepts from past designs together to create new concept variants. The design knowledge is captured through an empirical study that involves systematic dissection of various products and the construction of an online design repository. The method presented in this paper will help designers more effectively create design concepts through a computational search process. The grammar affords a representation of the design space as a tree of solutions built from an initial function structure as shown in Figure 18. Each transition in the tree corresponds to an application of a rule, thus incrementally building a final concept which is represented as one of the leaves of the tree. At the end, a search process returns different concepts for the same functional specifications with potentially varying degrees of complexity.

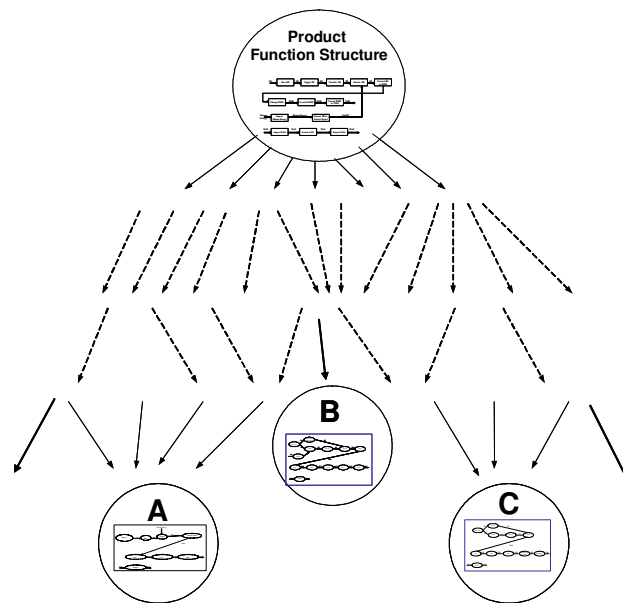


Figure 18: A visualization of the search tree in building a concept variant from a function structure of a product. Complete concepts will be arrived at the leaves of the tree.

There are various advantages of the approach developed here. First of all, 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. This is unlike other computational methods where a large number of complete solutions is generated first and pruned later based on a feasibility criteria. Secondly, potential solution bias is eliminated. Designers with varying levels of experience in different domains usually rely on solutions derived from domains where they feel the strongest. By exploring a vast space of past design solutions, our method can generate design configurations by combining solution principles from different domains. Thirdly, the open-endedness of a grammar formulation enables the synthesis of multiple input multiple output systems (MIMO) and accounts for function and component sharing. Finally, the likelihood of success in design is increased by use of our method, simply because the methodology itself is based on leveraging the successes of the past.

Future work is aimed at expanding our rule set. Currently we are studying other products to increase the number of components and solution principles in our knowledge base. Also, we are exploring ways to develop an evaluation method, which will allow a designer to sort or rank generated design concepts. Traditionally, computational design tools require very detailed specifications of part shapes and dimensions in order to evaluate certain performance parameters. For example, in order to determine a design's strength or rigidity a finite-element analysis must be performed on the components. While the results are accurate, they require defined part shapes and are time-consuming to perform. We are investigating "first pass" computational evaluations of the feasible concepts based on higher-level metrics that does not demand specific details of component geometry. Currently, we are considering such evaluation criteria as overall design complexity, potential failure modes, statistical likelihood of design success, and ease of assembly. Moreover, our research is exploring ways to evaluate the dynamics of the generated design configurations. This more rigorous evaluation of design concepts requires parameterization of component geometry. This would allow the analysis of potential system behavior by means of dynamic simulation. Towards that goal, we are exploring ways to integrate our synthesis tool with another computational tool (A-ODDS (Analytical Optimal Design of Dynamic Systems), Wu et al., 2005), which is being developed in our research lab. The A-ODDS systematically derives the dynamics model of a system by aggregating its sub-component models using a bond-graph representation. Then, it applies optimization methods to find out the best parameter setting for the design. We believe that this implementation and integration will not only improve our evaluation scheme, but it will also increase the scope of our automated synthesis method by moving the generated concepts closer to their final embodied states.

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, we are developing a graphical representation called the design assembly model (DAM) (Rajagopalan, 2005) that would complement configuration flow graphs (CFGs) and facilitate the addition of structural components and interfaces into the conceptual design configurations. Finally, since the library of components would ideally be based on a dynamic online repository, the creation of rules would also ideally be created dynamically in real time. The current set of 170 rules has been created through the careful deliberation that is typical of creating any grammar rule set. So, an interesting extension to the work presented here would be a method to develop these rules automatically.

## 9 References

- Bohm, M. and Stone, R., "Product Design Support: Exploring a Design Repository System", Proceedings of IMECE'04, IMECE2004-61746, Anaheim, CA, 2004.
- Bryant, C., Stone, R., McAdams, D., Kurtoglu, T., Campbell, M., 2005, "A Computational Technique for Concept Generation". Proceedings of ASME 2005 International Design Engineering Technical Conference, Long Beach, CA.
- Bracewell, R.H., and Sharpe, J.E.E., "Functional Descriptions Used in Computer Support for Qualitative Scheme Generation—Schemebuilder," AIEDAM, Vol. 10, No. 4, 1996, pg. 333-346.
- Cagan, J., 2001, "Engineering Shape Grammars," Formal Engineering Design Synthesis, Antonsson, E. K., and J. Cagan, eds., Cambridge University Press.

Campbell, M., J. Cagan and K. Kotovsky, 2000, "Agent-based Synthesis of Electro-Mechanical Design Configurations," *Journal of Mechanical Design*, Vol. 122, No. 1, pp. 61-69.

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, No. 102, pp. 137-151.

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

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

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

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

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.

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

Liang, V., C.J.J. Paredis, 2004, "A Port Ontology for Conceptual Design of Systems", *Journal of Computing and Information Science in Engineering*, Vol 4., Sept 2004.

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.

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.

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

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

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

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

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.

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.

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

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):102-125.

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

- Shooter, S., Keirouz, W., Szykman, S., Fenves, S, "A Model for Information Flow in Design", ASME Design Engineering Technical Conference Proceedings, Baltimore, MD, 2000.
- Sikand, A. and Terpenney, J., "A Web-Based Environment for Managing Product Knowledge", International Symposium on Collaborative Technologies and Systems (CTSO 4), San Diego, CA, 2004.
- 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, 2004, Salt Lake City, UT, DETC04/DTM-57130.
- 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, 1980a, "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.
- Strawbridge, B., McAdams, D. and Stone, R., 2002, "A Computational Approach To Conceptual Design," Proceedings of DETC2002, DETC2002/DTM-34001, Montreal, Canada.
- Suh, N., 1990, The Principles of Design, Oxford University Press.
- Summers, J.D., J. Shah., 2003, "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 DETC2002, DETC2002/CIE-34463, Montreal, Canada.
- Terpenney, J. and Mathew, D., "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, 2004.
- Terpenney, 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., June 25-27, 1997, pg. 69-80.
- Ullman, D., 1997, The Mechanical Design Process 2nd ed., McGraw-Hill.
- Ulrich, K. and Eppinger, S., 1995, Product Design and Development, McGraw-Hill.
- Umeda, Y. and Tomiyama, T., 1997, "Functional Reasoning in Design," IEEE Expert, March-April, pp. 42-48.
- Welch, R. V., and Dixon, J., 1994, "Guiding Conceptual Design Through Behavioral Reasoning," Research in Engineering Design, Vol. 6 pp. 169-188.
- Wu Z., B. R. Fernández, M. Campbell, 2005, "Probabilistic strategy based dynamic system design using bond graph and genetic algorithm," International Conference of Bond Graph Modeling, New Orleans, LA.