



Automated synthesis of electromechanical design configurations from empirical analysis of function to form mapping

Tolga Kurtoglu & M. I. Campbell

To cite this article: Tolga Kurtoglu & M. I. Campbell (2009) Automated synthesis of electromechanical design configurations from empirical analysis of function to form mapping, Journal of Engineering Design, 20:1, 83-104, DOI: [10.1080/09544820701546165](https://doi.org/10.1080/09544820701546165)

To link to this article: <https://doi.org/10.1080/09544820701546165>



Published online: 09 Feb 2009.



Submit your article to this journal [↗](#)



Article views: 333



View related articles [↗](#)



Citing articles: 5 View citing articles [↗](#)

Automated synthesis of electromechanical design configurations from empirical analysis of function to form mapping

Tolga Kurtoglu* and M.I. Campbell

Department of Mechanical Engineering, The University of Texas at Austin, Austin, Texas, USA

(Received 12 January 2007; final version received 11 May 2007)

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 artefact 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 artefacts in the form of grammar rules. From an initial implementation of 189 rules extracted from 23 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

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 subfunctions at lower levels of abstraction. Solutions to these subfunctions are then sought and synthesised 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 subfunctions such as brainstorming, mind mapping, directed and undirected searches, and morphological analysis. However, most of these are geared towards broadening and organising solutions to subfunctions and provide little or no assistance in synthesising 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 subfunctions of the design are listed in different columns. As a whole, the morphological matrix captures the potential solution

*Corresponding author. Email: tolga@mail.utexas.edu

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	electrolytic goo	
supply electrical energy	battery	capacitor	electrolytic 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			

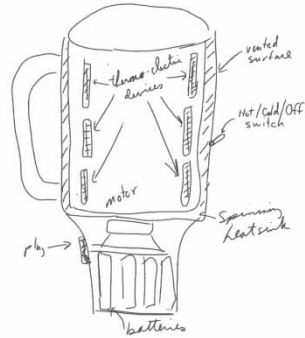


Figure 1. Morphological matrix for a ‘thermal mug’ design and a conceptual sketch derived from it.

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 subfunction?
- How should the selected subsolutions of a concept variant be synthesised together into a final configuration?

These questions summarise 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 artefact 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 synthesise 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 rigour 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 17 consumer products, 189 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 the 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). We then 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 summarise 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 subfunctions, solutions to which are then synthesised 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 behaviour information to loose hierarchical artefact 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 – Artefacts, Functions, Forms, Behaviours and Flows – which contain information relative to their denoted name. Similar function-based representations have been proposed by other workers to support search and modelling processes of conceptual design (Terpenney 1997, Terpenney and Mathew 2004, Sikand and Terpenney 2004). Japanese researchers have also explored a consistent language for describing the functionality of products and relating it to product behaviour (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 (1999) that 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 present 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 utilises a function–component matrix (FCM) and a filter matrix to generate a morphological matrix of solutions for functions in a conceptual functional model. The FCM uses columns of components and rows of functions to characterise 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 FCMs to generate a matrix describing the complete solution set. Following Strawbridge *et al.*’s work, Bryant *et al.* (2005)

Table 1. Flow classes and their basic categorisations.

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

developed an automated concept generation tool that utilises 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 function-based design synthesis approach is that of Xu *et al.* (2006). This research develops a design reuse framework that combines product modelling, knowledge extraction, and a computational search for concept generation and evaluation.

Table 2. Function classes and their basic categorisations.

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

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. Other notable computational tools developed for concept generation are the agent-based system presented in the A-Design research (Campbell *et al.* 2000) and the catalogue design method used in Chakrabarthi and Bligh (1996). 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 behaviour; Paynter 1961) with the design methodology posed by Pahl and Beitz. The representation developed here aims to relate components in terms of their dynamic behaviour, 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 and Sharp (1996).

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 (Pinilla *et al.* 1989, Fu *et al.* 1993, Schmidt and Cagan 1995, 1998, Li *et al.* 2001, 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 artefact. 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 of 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 1984, 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 organising design knowledge are studying existing artefacts, recording and storing design knowledge in the repository, and the development of standardised 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 emphasise concrete experience with the existing product. We strive to fully understand the product in terms of functionality, components, product hierarchy, manufacturing, and assembly. Each



Figure 2. Dissection of a Black & Decker hand-held vacuum cleaner.

Table 3. The 23 products used as an empirical basis for the development of our method.

Product 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	Wine Opener
Knife Sharpener	Paper Shredder
Fruit Peeler	Drink Mixer
Pencil Sharpener	

product is completely dissected and documented. As the product is disassembled, a bill of materials is constructed to document the components. Each component is labelled and photographed as it is removed from the product, and data recorded in a tabular format listing its attributes, such as the assembly name, part number, quantity, part description, part function(s), input–output flows, physical parameters and predicted manufacturing processes.


In this study, 23 consumer products were 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. Domain here is defined based on the fundamental flow types (input, output, and intermediate flows) that govern the functioning of a product. 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 for the development of our method, we aimed for comprehensive coverage of fundamental flow types presented in Table 1. Figure 2 shows one of these products, a hand-held vacuum cleaner, at various stages of teardown. The complete list of products that were analysed is presented in Table 3.

3.2. Design knowledge repository

The web-based design knowledge repository (<http://function.basiceng.umn.edu/repository>), managed at the University of Missouri-Rolla, serves as a forum to store and organise the design knowledge derived from product teardowns. The design data are recorded in the repository using a FileMaker template. Following entry and storage into FileMaker, a platform-independent data set is output as XML. By creating Java Server Pages, the XML from the database server can be viewed as HTML through a standard web browser (Bohm and Stone 2004) as shown in Figure 3.

Bottom Case Assembly
Motor Assembly
Slicer
blade fixtures
blade guide/cover
blades
bottom case
column 1
crimp
gear 1
glue
impeller
locking plate
motor
plug and cord
release button
retaining clips
rivet
safety pin
screw 1
screw 2
screw 3
screw 4
screw 5
screw 6
solder
switch plate

System: Black and Decker - Slice Right

Artifact Name	gear 1	Artifact Photo
Part Family	not specified	 <p>click on image for full size</p>
Part Number	10	
Sub Artifact Of	motor assembly	
Quantity	1	
Description	nylon gear with cam additions on either side	
Artifact Color	maroon	
Component Naming	gear	

Input Artifact	Input Flow	Subfunction	Output Flow	Output Artifact
motor	mechanical energy	change	mechanical energy	internal
internal	mechanical energy	transfer	mechanical energy	blade fixtures

Supporting Functions

blade fixtures	solid material	guide	solid material	internal
----------------	----------------	-------	----------------	----------

Physical Parameters Manufacturing Process

width	0,58	material	nylon
radius	0,37	no process specified	

Figure 3. UMR Design Repository web interface.

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 fulfil them.

3.3. Design knowledge representation

To derive uniformity and consistency in representing design knowledge, we have adopted two standardised vocabularies. At a functional level, we use the aforementioned functional basis language developed by Hirtz *et al.* (2002). This standardised lexicon of terms defines a comprehensive set of function and flow names at three levels of abstraction to support both high-level and low-level functional modelling independent of the physical structure of the artefact. 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 135 distinct component names. Table 4 presents a partial list of the component basis.

Our methodology is built upon a framework that represents design knowledge using graphs. This facilitates implementation of effective graph manipulation methods (graph grammars, constraint satisfaction, etc.) for automated generation of design concepts. In this research, we utilise two graph-based representations: *function structures* and *configuration flow graphs* (CFGs). In the following sections we summarise these two graph representations.

3.3.1. Representation of function: function structure

A function structure is a graphical representation of the decomposition of the overall function of a product into smaller, more elemental subfunctions. The subfunctions are connected by flows (of types energy, material and signal) on which they operate. Overall, a function structure represents the transformation of input flows into output flows at the system level. 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 repeatable formation of a function structure. In this research, we use the aforementioned functional basis as a standard vocabulary for the construction of function structures (McAdams *et al.* 1998, Kurfman *et al.* 2001). To illustrate an example, the function structure of an electric toothbrush is shown in Figure 4a.

3.3.2. Representation of form: configuration flow graph

A 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

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	balloon, 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.

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 behaviour of components by modelling them as ‘black box’ entities that transform certain input flows into certain output flows.

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. Figure 4b shows an example of a CFG for an electric toothbrush product accompanied by a conceptual sketch. As can be seen from the figure, components that are present in a design, their connectivity, and physical interfaces between a design’s components can be captured graphically using a configuration flow graph.

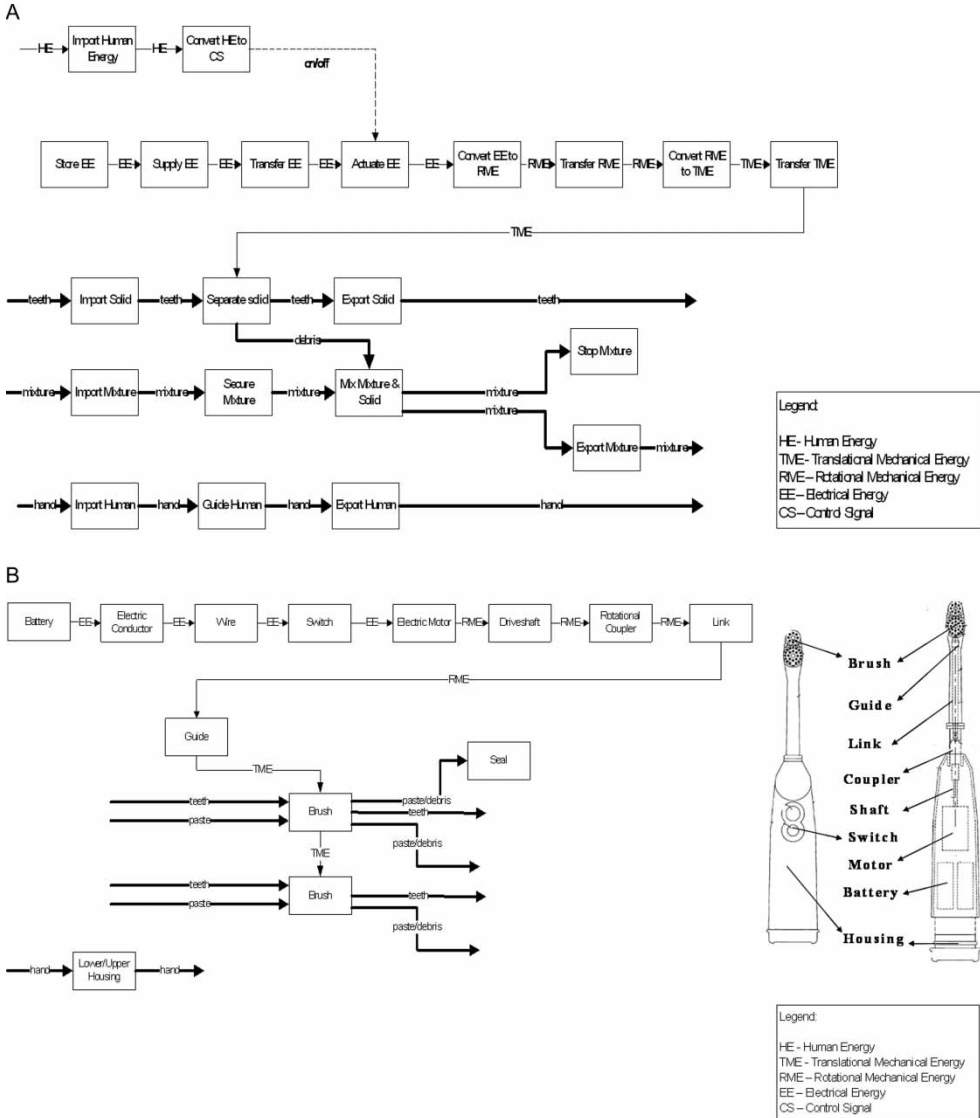
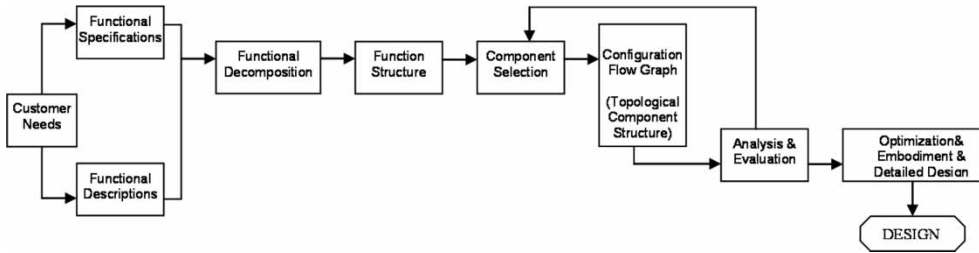


Figure 4. (a) Function structure of an electric toothbrush. (b) CFG of an electric toothbrush.

4. Research approach

In this section, we present our approach for creating new design configurations from functional requirements. We apply leverage to the expanding online library of products and we derive design rules from it to capture the knowledge of the original designer that is employed in the construction of conceptual solutions. In creating the rules, we observe the common uses of the components in past designs and formulate this knowledge as design ‘grammar rules’. Accordingly, during product teardowns we capture an existing product’s CFG and its function structure. We then use this graph database to formulate our graph grammar language that encodes the mapping from the functional space to the configuration space.



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 research follows the overall process flow of function-based synthesis shown in Figure 5. By following this scheme, a design is changed from an abstract set of needs to the embodiment of the final design. Our computational method pursues this process as we automatically search for concept variants. We initiate the design process at the level of *a function structure* or *a functional model* (Pahl and Beitz 1988), which is a form-independent expression of the designed product. The output of the process is *a topological component structure* where specific electromechanical components are first associated with individual or sets of subfunctions from the function structure and then composed into a design configuration based on their physical interactions. Feasibility and consistency are maintained throughout the design process while transitioning between these abstraction levels.

Our synthesis methodology is based on constructing a CFG given the function structure for a conceptual design. In this regard, our approach is unique 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, our 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 modelled after basic grammar conventions where rules are comprised of a left-hand side and a right-hand side. The left-hand side contains the state that must be recognised in the function structure, and the right-hand side shows how the design is updated to a new configuration. Consider Figure 6 as an example. This rule states that if functions 1, 2 and flows 1–4 are recognised in the input function structure, then the design can be updated by the addition of components 1 and 2 and their physical interactions represented by flows 1–3.

In detail, the transformation from the function structure to CFG is part of the recognise–choose–apply cycle. Given an initial function structure, S , we first recognise all possible rules that have their left-hand side as a subset of S . This recognition step is followed by choosing 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

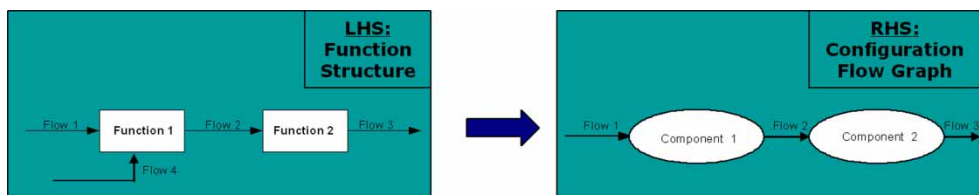


Figure 6. A hypothetical rule.

the instructions provided in the right-hand side. 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 subfunction or set of subfunctions.

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 fulfil various functions.

One of challenges we faced in our study was determining the level of granularity of the function structures created for analysed products. In the functional basis, three levels of abstraction are developed to represent functions and flows: primary, secondary, and tertiary. For formalising our grammar definition we have adopted the secondary level. According to this, functional modelling of the products is performed using only function and flow terms presented in Tables 1 and 2. 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 the 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

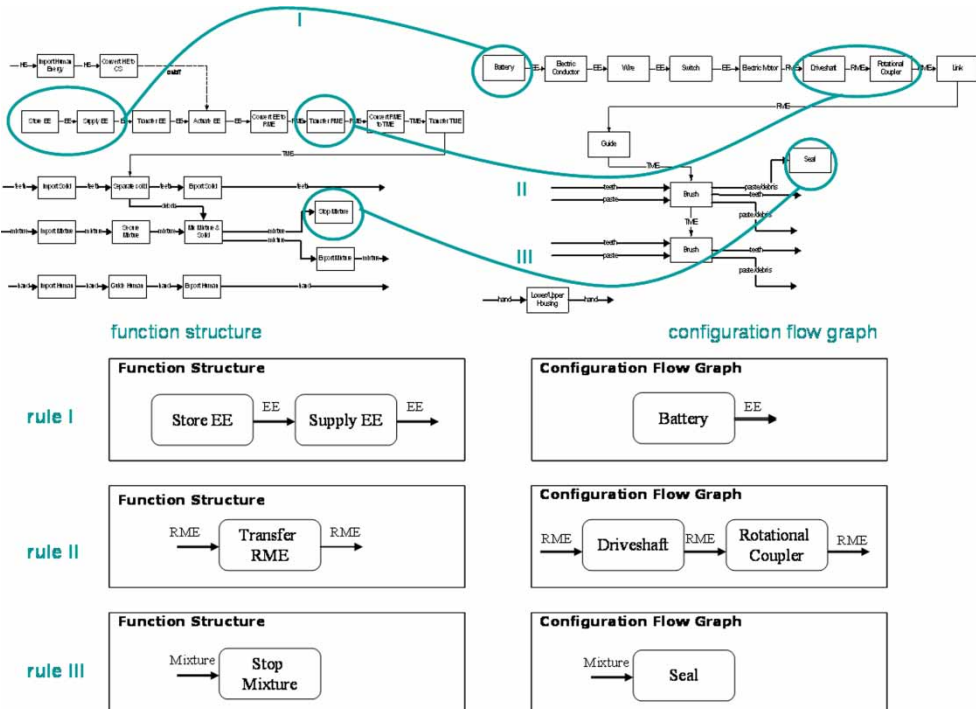


Figure 7. Illustration of rule derivation (only three rules) from empirical analysis of a toothbrush product.

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 recognises ‘store electrical energy’ and ‘supply electrical’ and their associated flows in the function structure. If the two functions and three flows are recognised, it adds the component ‘battery’ to the CFG at the appropriate location. Similarly, the second shown in Figure 7 recognises the function ‘transfer rotational mechanical energy’ in the function structure and adds a ‘driveshaft’ and a ‘rotational coupler’ to the design configuration.

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 standardised 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 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 family of products we have chosen to analyse. Our current set of rules does not fully represent the entire set of engineering artefacts; however, it provides a framework for the method to be extended to other product families or to other design contexts. Currently, we have 189 rules derived from 23 products. The complete list of 189 rules is shown in detail online (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.

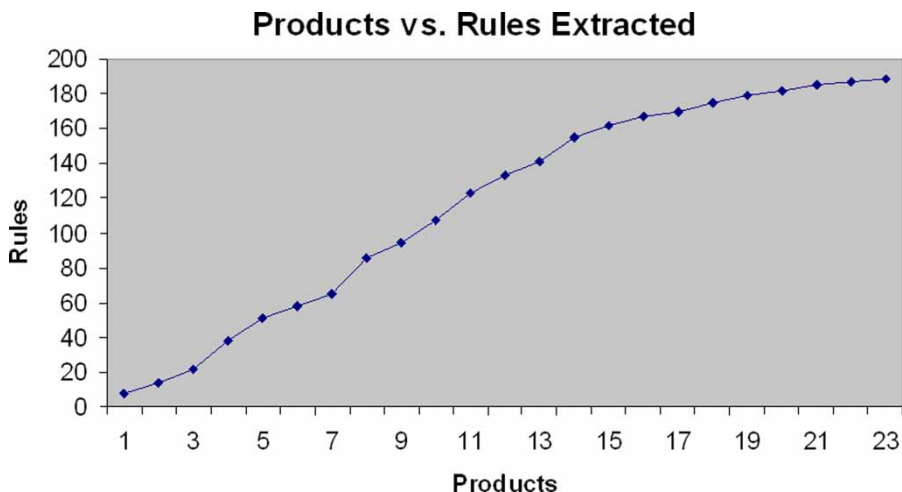


Figure 8. Graph of the number of products examined and the rules obtained from each of them.

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.

The algorithm first reads the function structure and replaces subfunctions with components after application of each new rule until no further rules can be recognised. Figure 10 shows this process. At the beginning, none of the subfunctions 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 subfunctions in the function structure designate the subfunctions that are mapped to a component solution. The result of this process is the completed design configuration shown at the end of the figure.

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

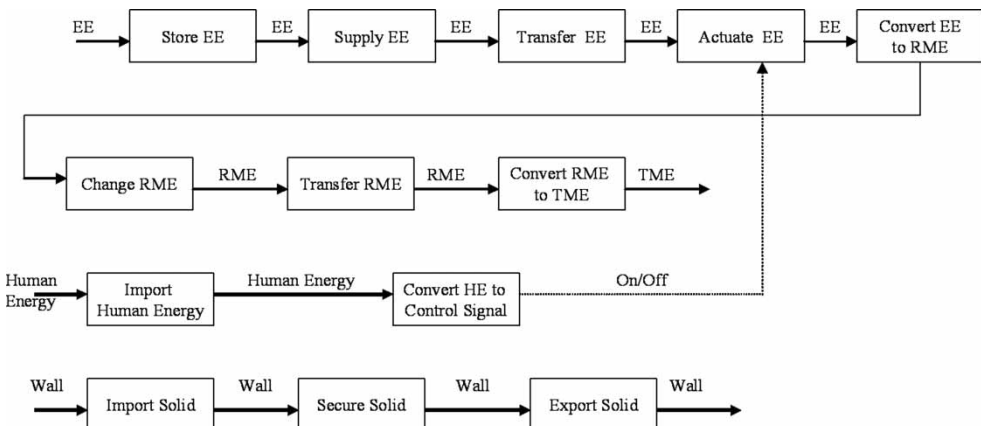


Figure 9. Function structure for the wall climber toy design problem.

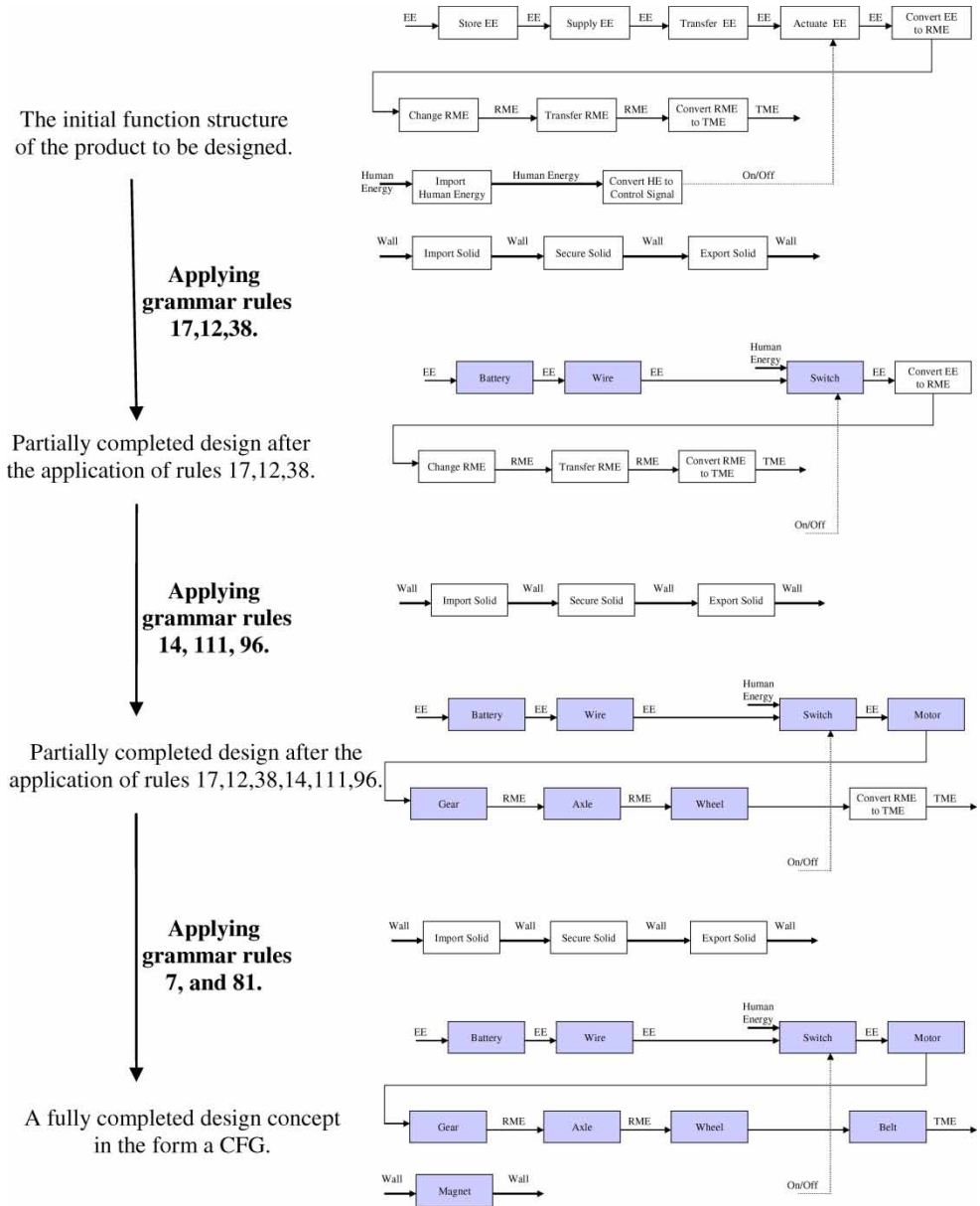


Figure 10. A pictorial representation of building the CFG a from function structure using the rule set.

toothbrush (battery), an electric knife (wire) and a hand-held vacuum cleaner (switch). 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 synthesising component solutions together that have been successfully used in the design of past products.

Compatibility has to be ensured while synthesising 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 subsolutions 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 ‘recognise’ 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 practical feasibility.

6.1. Experimental study

To further assess the effectiveness of our automated concept generator as a design aid, we 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 11 shows an example morphological matrix and a design concept synthesised from it by a group of students. The highlighted subsolutions of the morph matrix are included in the final sketched concept.

For this group, the results show that students using a traditional method created very similar concepts (a total of three) to those generated by the execution of grammar rules. Consider the sketched concept variant of Figure 11. It has eight 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 manoeuvre 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.

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	

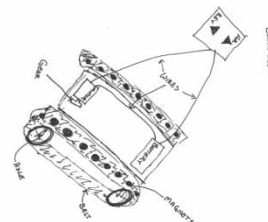


Figure 11. A morphological matrix and a concept sketch drafted from it.

7. Implementation

An algorithm has been created and programmed in C# using Microsoft Visual Studio that follows the flowchart shown in Figure 12. 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-based and diagram-based applications. Our automated concept generation software uses Netron's standard tools to provide designers a natural visual interface.

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 13 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.

The program then executes a recognise–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 189 rules and checking each for applicability. Any rule that satisfies recognition conditions is then listed as a 'recognised' rule along with its corresponding location.

Recognition is followed by the selection of the rule to be applied. In our 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 spans the design space for all possible configurations. Specifically, the computer employs a modified breadth-first search algorithm, which includes a filtering mechanism that removes duplications of previously visited nodes from the search space. The result of this search process is a set of complete design configurations (CFGs) that are unique and that consist of different selection and configuration of component concepts. 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.

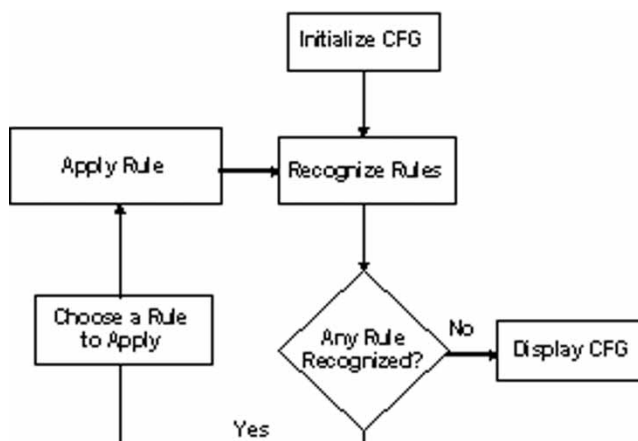


Figure 12. Detailed flowchart of rule processing.

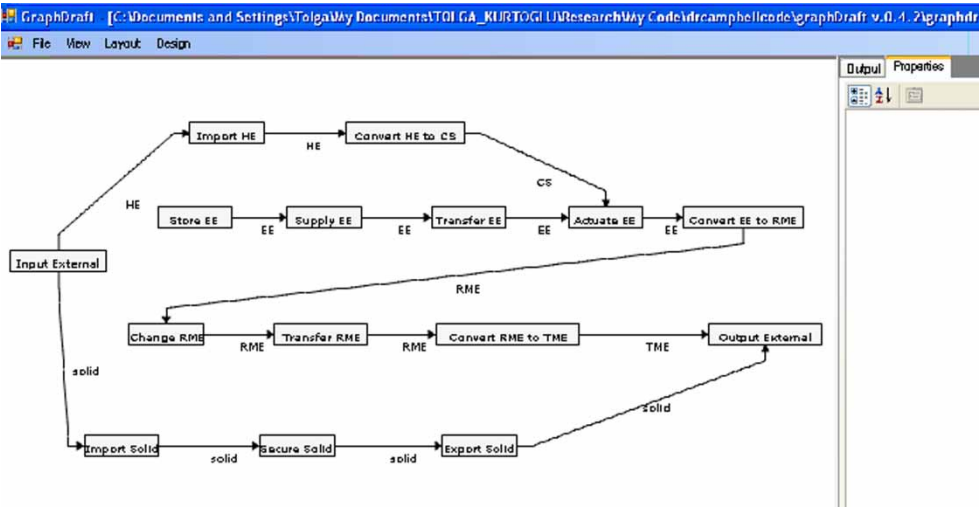


Figure 13. Graphical user interface of the developed automated concept generation software. The designer sketches the function structure of the system that is to be designed.

The third and final step of rule processing is applying a selected rule. After the program makes a selection from the recognised rules list, it updates the function structure graph to a new configuration by replacing related subfunctions with their associated component(s) as described by the selected rule. This ensures that a subfunction is not recognised again, once it has been assigned a component solution. Figure 14 shows the state of the graph, after the application of ‘wire’ and ‘battery’ rules. In intermediate states, such as the one shown in Figure 14, the design is only partially completed. Therefore, the graph in these states is a hybrid graph consisting of both function structure and CFG elements.

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.

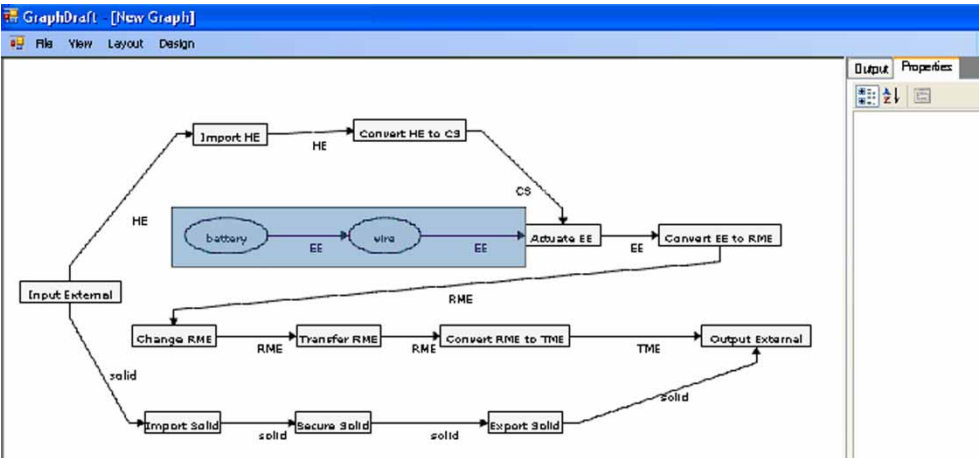


Figure 14. Screenshot of the user interface at a partially completed design state.

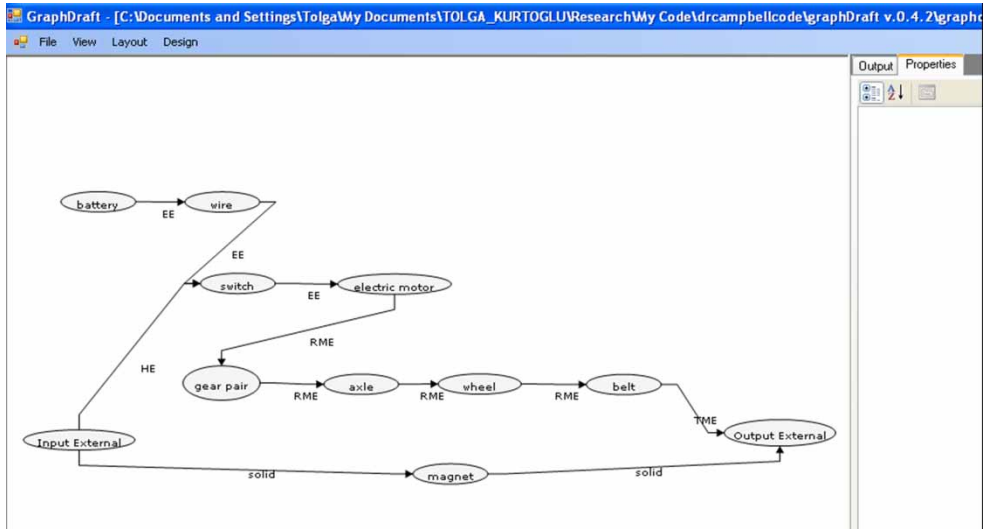


Figure 15. Screenshot of the user interface after completion of design.

The program manages the rules and their applications until no further rules can be applied, thus terminating when complete design configurations are built for the given functional description. For the ‘wall climber toy example’ of Section 6, the computer generates 1188 unique configurations using the grammar. One of these CFGs is shown in Figure 15.

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 automatically 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 16. Each transition in the tree corresponds to an application of a rule, thus incrementally building a final concept that is represented as one of the leaves of the tree. During this tree traversal, the computer systematically selects the rules to be applied in order to search for potential design configurations. The basis and the guidelines to select the rules are embedded in the breadth-first search algorithm. Accordingly, each applicable grammar rule is selected by the computer with equal likelihood as the tree is traversed. An exhaustive search algorithm is selected here, because the authors believe that finding a good conceptual solution requires an in-depth search of the design space and often necessitates generation of as many design alternatives as possible. At the end, the search process returns different concepts for the same functional specifications with potentially varying degrees of complexity.

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 are generated first and pruned later based on a feasibility criterion. Secondly,

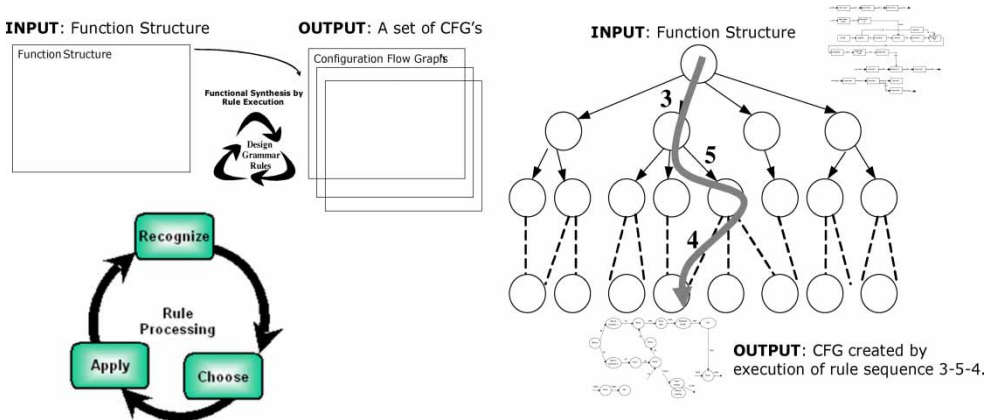


Figure 16. A visualisation of the search tree in building a concept variant from a function structure of a product. Complete concepts will be arrived at the leaves of the tree.

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, 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, other products are being studied to increase the number of components and solution principles in our knowledge base. Also, ways to develop an evaluation method that will allow a designer to sort or rank generated design concepts are explored. 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. Under investigation are 'first pass' computational evaluations of the feasible concepts based on higher-level metrics that do not demand specific details of component geometry. Currently considered are 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 parameterisation of component geometry. This would allow the analysis of potential system behaviour by means of dynamic simulation. Towards that goal, we are exploring ways to integrate our synthesis tool with another computational tool – Analytical Optimal Design of Dynamic Systems (A-ODDS; Wu *et al.* 2005) – which is being developed in our research laboratory. The A-ODDS systematically derives the dynamics model of a system by aggregating its subcomponent models using a bond-graph representation. Then, it applies optimisation methods to find the best parameter setting for the design. This implementation and integration will not only improve the 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 (Rajagopalan *et al.* 2005) that would complement 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 189 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 by leveraging graph theoretic algorithms that search for the intersection of function structures and CFGs.

References

- Bohm, M. and Stone, R., 2004. Product design support: exploring a design repository system. In: *Proceedings of ASME international mechanical engineering congress and exposition*, 13–20 November 2004. Anaheim, CA. IMECE2004-61746.
- Bracewell, R.H. and Sharpe, J.E.E., 1996. Functional descriptions used in computer support for qualitative scheme generation – Schemebuilder. *AIEDAM*, 10 (4), 333–346.
- Bryant, C., Stone, R., McAdams, D., Kurtoglu, T. and Campbell, M., 2005. A computational technique for concept generation. In: *Proceedings of ASME 2005 international design engineering technical conference*, September 24–28 2005, Long Beach, CA.
- Cagan, J., 2001. Engineering shape grammars. In: E. K. Antonsson and J. Cagan, eds. *Formal engineering design synthesis*. New York: Cambridge University Press, 65–93.
- Campbell, M., Cagan, J. and Kotovsky, K., 2000. Agent-based synthesis of electro-mechanical design configurations. *Journal of mechanical design*, 122 (1), 61–69.
- 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, 313–331.
- Fu, Z., De Pennington, A. and Saia, A., 1993. A graph grammar approach to feature representation and transformation. *International journal of computer integrated manufacturing*, 6 (102), 137–151.
- Greer, J., Stock, M.E., Stone, R. and Wood, K., 2003. Enumerating the component space: first steps towards a design naming convention for mechanical parts. In: *Proceedings of ASME international design engineering technical conference*, September-6, Chicago, IL. DETC2003/DTM-48666.
- 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*. Berlin: Springer-Verlag.
- Kitamura, Y. and Mizoguchi, R., 1998. Functional ontology for functional understanding. In: *Twelfth international workshop on qualitative reasoning (QR-98)*, May 26–29, Cape Cod, MA. AAAI Press, 77–87.
- Kurtoglu, T., Campbell, M., Bryant, C., Stone, R. and McAdams, D., 2005. Deriving a component basis for computational functional synthesis. In: *Proceedings of international conference on engineering design, ICED'05*. 15–18 August 2005, Melbourne, Australia.
- Li, X., Schmidt, L., He, W., Li, L. and Qian, Y., 2001. Transformation of an EGT grammar: new grammar, new designs. In: *Proceedings of ASME 2001 design engineering technical conference*, 9–12 September 2001, Pittsburgh, PA. DETC2001/DTM-21716.
- Little, A., Wood, K. and McAdams, D., 1997. Functional analysis: a fundamental empirical study for reverse engineering, benchmarking and redesign. In: *Proceedings of the ASME international design engineering technical conferences*, 14–17 September 1997, Sacramento, CA. 97-DETC/DTM-3879.
- Murdock, J., Szykman, S. and Sriram, R., 1997. An information modeling framework to support design databases and repositories. In: *Proceedings of ASME international design engineering technical conference*, September 14–17 1997, Sacramento, CA. DETC97/DFM-4373.
- Otto, K. and Wood, K., 2001. *Product design: techniques in reverse engineering and new product development*. New Jersey: Prentice-Hall.
- Pahl, G. and Beitz, W., 1988. *Engineering design: a systematic approach*. Berlin: Springer-Verlag.
- Paredis, C.J.J., Diaz-Calderon, A., Sinha, R. and Khosla, P.K., 2001. Composable models for simulation-based design. *Engineering with computers*, 17, 112–128.
- Paynter, H.M., 1961. *Analysis and design of engineering systems*. Cambridge, MA: MIT Press.
- Pinilla, J.M., Finger, S. and Prinz, F.B., 1989. Shape feature description using an augmented topology graph grammar. In: *Proceedings of the NSF engineering design research conference*, 11–14 June 1989 Amherst, MA. pp. 285–300.
- Rajagopalan, V., Bryant, C., Johnson, J., Stone, R., McAdams, D., Kurtoglu, T. and Campbell, M., 2005. Creation of assembly models to support automated concept generation. In: *Proceedings of ASME 2005 International design engineering technical conference*, September 24–28 2005, Long Beach, CA.
- Rozenberg, G., ed., 1997. *Handbook of graph grammars and computing by graph transformation—volume 1: foundations*. Singapore: World Scientific.
- Sasajima, M., Kitamura, Y., Ikeda, M. and Mizoguchi, R., 1995. FBRL: a function and behavior representation language. In: *Proceedings of international joint conference on artificial intelligence IJCAI'95*, 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*, 120, 2–9.
- Shooter, S., Keirouz, W., Szykman, S. and Fenves, S., 2000. A model for information flow in design. In: *Proceedings of ASME international design engineering technical conference proceedings*, 10–13 September Baltimore, MD.
- Sikand, A. and Terpenney, J., 2004. A web-based environment for managing product knowledge. In: *International symposium on collaborative technologies and systems (CTSO 4)*, San Diego, CA.
- Sridharan, P. and Campbell, M.I., 2004. A grammar for function structures. In: *Proceedings of ASME 2004 international design engineering and technical conference and computers and information in engineering conferences*, 28 September–2 October 2004, Salt Lake City, UT. DETC04/DTM-57130.
- Starling, A.C. and Shea, K., 2005. Virtual synthesizers for mechanical gear systems. In: *Proceedings of international conference on engineering design ICED'05*, Melbourne, Australia.
- Stiny, G., 1980. Introduction to shape and shape grammars. *Environment and planning B: planning and design*, 7, 343–351.
- Stone, R. and Wood, K., 1999. Development of a functional basis for design. In: *Proceedings of AMSE international design engineering technical conference*, 12–15 September 1999, Las Vegas, NV. DETC99/DTM-8765.
- Strawbridge, B., McAdams, D. and Stone, R., 2002. A computational approach to conceptual design. In: *Proceedings of ASME international design engineering technical conference*, 29 September–02 October 2002, Montreal, Canada. DETC2002/DTM-34001.
- Suh, N., 1990. *The principles of design*. Oxford University Press.
- Szykman, S., 2002. Architecture and implementation of a design repository system. In: *Proceedings of ASME international design engineering technical conference*, 29 September–02 October 2002, Montreal, Canada. DETC2002/CIE-34463.
- Terpenney, J., 1997. An integrated system for functional modeling and configuration in conceptual design. In: *Proceedings of the seventh international flexible automation and intelligent manufacturing conference*, 25–27 June 1997 Middlesbrough, UK. pp. 69–80.
- Terpenney, J. and Mathew, D., 2004. Modeling environment for function-based conceptual design. In: *Proceedings of ASME international design engineering technical conferences and computers and information in engineering conferences*, 28–30 September 2004, Salt Lake City, UT.
- Ullman, D., 1997. *The mechanical design process* 2nd edn. New York: McGraw-Hill.
- Ulrich, K. and Eppinger, S., 1995. *Product design and development*. New York: McGraw-Hill.
- Umeda, Y. and Tomiyama, T., 1997. Functional reasoning in design. *IEEE expert*, March–April, 42–48.
- Welch, R.V. and Dixon, J., 1994. Guiding conceptual design through behavioral reasoning. *Research in engineering design*, 6, 169–188.
- Wu, Z., Fernández, B.R. and Campbell, M., 2005. Probabilistic strategy based dynamic system design using bond graph and genetic algorithm. In: *Proceedings of international conference of bond graph modeling and Simulation*, 23–37 January 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*, 17, 27–44.