INTERNATIONAL DESIGN CONFERENCE - DESIGN 2006
Dubrovnik - Croatia, May 15 - 18, 2006.

DESIGN 2006

A GRAPH GRAMMAR BASED FRAMEWORK FOR AUTOMATED CONCEPT GENERATION

T. Kurtoglu and M. I. Campbell

Keywords: concept generation, graph grammars, design reuse

1. Introduction

Computer-aided tools are widely utilized at the detailed stages of product development thanks to the formal representations developed to support modeling, simulation and analysis. Few computational tools are available, however, to assist designers during the conceptual phase of design. The following factors are recognized as the primary contributors for the existence of this gap: (1) the intrinsic difficulty in computationally representing the "highly abstract" design knowledge that is relevant to conceptual design, and (2) the lack of a formal framework that enables the integration of this knowledge for modeling and designing at the early stages. This paper presents a method that is developed to address the aforementioned challenges and to computationally support the conceptual design phase of product development.
In this research, we develop a methodology and resulting representations and computational tools to compute design alternatives. Unlike other research that attempts to automatically synthesize a design solution, this research leverages an expanding online repository of electromechanical products from which design knowledge is systematically extracted and integrated into a graph-grammar-based framework. The framework includes representations that capture designs at two levels of abstraction (the functional level and the component level) and, it allows formulation of the design knowledge as grammar rules that capture the mapping between these two levels. The rules created from the repository are included in a computational search process that works with a designer in navigating the design space to create conceptual design configurations from detailed specifications of product function. The configuration description includes the choice of components and their topological structure describing the connectivity of components and the physical interfaces that result. Using the developed design tools, a design team can generate multiple and feasible configurations, which then can automatically be evaluated and ranked based on a variety of design objectives.

2. Review of Related Work

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 developed and synthesized together to arrive at a final form of a product. Our computational approach to concept generation follows this function-based synthesis method. One of the critical requirements for the function-based synthesis approach is the representation of functional knowledge. Several researchers have developed systems to capture abstract engineering design models of functionality. 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 [Szykman, 2002]. Within the NIST model, the design information is organized using five sections: Artifacts, Functions, Forms, Behaviors and Flows. Similar to this effort, the concept of a functional basis is developed by Stone and Wood [1999] to capture product functionality. The functional basis includes a set of terms that span the space of all functions and all flows. Using this functional vocabulary, device function can be defined for a given system using a function structure.

Built upon this methodology, Bryant, et al. [2005] developed a concept generation technique that utilizes a repository of existing design knowledge and a set of matrix-manipulation algorithms. In this research, function-component matrices (FCM's) and component-component matrices are used to capture function to component mappings and physical compatibilities between components. An aggregated matrix, then, can be constructed from individual product matrices describing the complete solution space. Another two notable computational tools developed for concept generation are the agent-based system presented in the A-Design research [Campbell et al, 2000] and the catalog design method used in Chakrabarthi and Bligh [1996]. In both these approaches, input-output characteristics of component elements are used to synthesize a system level design.

In this research, we combine the formal function based synthesis method [Pahl and Beitz, 1988] with graph representations developed to facilitate the formulation of a graph grammar language for creating design configurations. The concept of a grammar is that a set of rules is constructed to capture a specific domain 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 [Cagan, 2001]. Such representations can produce a wider variety of candidates since solutions only need to have a common starting point. In this work, we use a functional description of a product as a starting point and seek multiple configuration solutions that address the functional requirements.

3. Research Fundemantals

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. Furthermore, we propose that an automated process can estimate the worth of such concept variants based on a number of evaluation criteria. The proposed research follows the overall process flow of function-based synthesis shown in Figure 1. 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 structur*e 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 sub-functions from the function structure and then composed into a design configuration based on their physical interactions. Feasibility and consistency is maintained throughout the design process while transitioning between these abstraction levels.

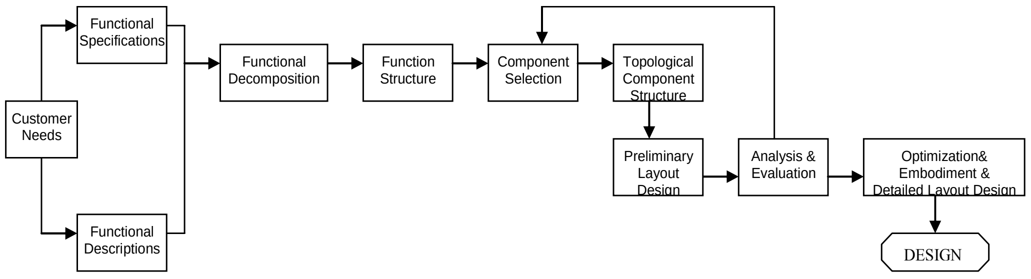
Functional Specifications
Customer Needs
Functional Descriptions
Functional Decomposition
Function Structure
Component Selection
Topological Component Structure
Preliminary Layout Design
Analysis & Evaluation
Optimization& Embodiment & Detailed Layout Design
DESIGN

Figure 1. Typical process flow of the function based synthesis method.

Our design method following the process flow of Figure 1 manifests itself as a computational design tool. The development of this design tool is based on the framework illustrated in the pyramidal structure of Figure 2. A pyramid is a useful visualization here, since each level builds upon the foundation of the lower levels and reduces the complexity of its foundation in order to perform its intended tasks. The primary motivation behind our method is to capture design knowledge from existing engineering artifacts, therefore the wealth of *existing engineering artifacts* constitute the bottom-most layer of the pyramid. The next layer is *design knowledge extraction and organization*. At this layer, design knowledge is gathered and organized using standardized vocabularies (taxonomies) and an online design knowledge repository. The middle layer is *graph representations of design,* where designs are represented at the functional and the component level. These graph representations form the skeleton of the graph grammar language that our concept generation method is built upon. At the top-most layer, resides the *computational synthesis method*. It is at this layer where concept variants are generated and evaluated based on graph grammar and search algorithms developed. In the following two sections, we further explain some of the research background that led to the development of our computational design tool.

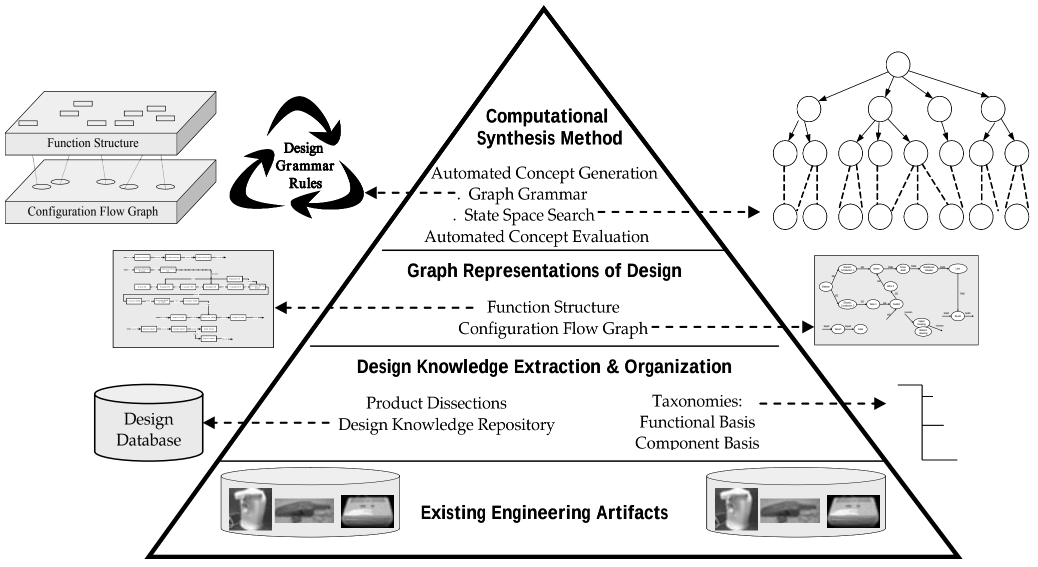
Function Structure
Configuration Flow Graph
Design Grammar Rules
Computational Synthesis Method
Automated Concept Generation
. Graph Grammar
. State Space Search
Automated Concept Evaluation
Graph Representations of Design
Function Structure
Configuration Flow Graph
Design Knowledge Extraction & Organization
Product Dissections
Design Knowledge Repository
Taxonomies:
Functional Basis
Component Basis
Design Database
Existing Engineering Artifacts

Figure 2. The framework of our computational synthesis method

3.1 Design Knowledge Extraction and Organization

The fundamental research question we address in extracting design knowledge is concerned with representing and storing the information in such a way that it can be reused for future product designs. This requires a systematic approach to the knowledge extraction as well as its organization so that it can be efficiently indexed, searched and retrieved. The first step in knowledge extraction is the dissection of products. (Examples include power tools, kitchen appliances and other household products) In this step, we emphasize concrete experience with the existing product. We analyze the product in terms of functionality, components, architecture, and manufacturing. Each product is completely dissected and documented by creating a bill of materials that lists each component of the product and its attributes such as part number, quantity, part functions, input-output flows, physical parameters, predicted manufacturing processes and potential failure modes. The design knowledge derived from product teardowns is stored and organized in a web-based design knowledge repository, managed at the University of Missouri-Rolla. The knowledge contained in the repository is steadily expanding and currently includes detailed information on approximately 70 consumer products with over 3700 components. The design data is recorded in the repository using a standard FileMaker Pro template, which can be 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]. Our research has leveraged the repository data to extract a design grammar language that captures the relationship between specific functions and the components that are used to fulfill them.

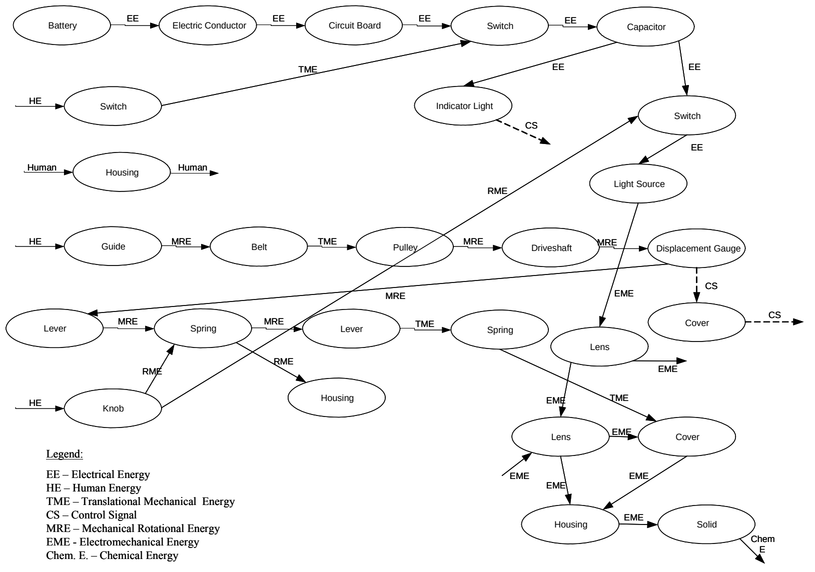

Figure 3. A configuration flow graph of a disposable camera

Deriving uniformity and consistency in representing the functional and component design knowledge is essential to the development of our method. Towards that goal, we have adopted two standardized vocabularies. At a functional level, we use the aforementioned *functional basis* language [Stone and Wood, 1999]. 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 a system. Using this functional vocabulary, we define device function for a given product by building a function structure. Similar to the functional basis in principle, we have also sought an accepted set of component names [Kurtoglu et al., 2005]. According to the *component basis*, each artifact is classified under a specific component name that can be thought of as the most generic abstraction representing a component concept. The basis allows for well-defined function-based groupings of artifacts to be used in the creation of various design representations and design tools. It also eliminates artifact redundancies that may not be immediately evident due to variations in user-dependent artifact naming. The current component basis contains 110 distinct fundamental component names. (Example basis names include: acoustic insulator, agitator, electric motor, bearing, battery, belt, gear, housing, screw, etc.)

3.2 Graph Representations of Design

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 utilize two graph-based representations: *function structure* and *configuration flow graph (CFG)*. Function structure is a graphical representation of the decomposition of the overall function of a product into smaller, more elemental sub-functions. The sub-functions are connected by flows (of types energy, material and signal) that they operate on. Overall, a function structure represents the transformation of input flows into output flows at the product level. A Configuration Flow Graph (CFG), on the other hand, is a graph representation of a design configuration. In a CFG, nodes of the graph represent product components, whereas arcs represent energy, material or signal flows between components. 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. At an abstract level, the CFG also represents the behaviour of components by modeling them as "black box" entities that transform certain input flows into certain output flows. Figure 3 shows an example of a CFG created for a disposable camera.

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.

4. Research Approach

In this section we present our approach for creating new design configurations from functional requirements. We leverage the expanding online repository of components and we derive design rules from it to capture the knowledge of the original designer that is employed during the creation of designs. In deriving the rules, we observe the common uses of the components in the repository and formulate this knowledge as "grammar rules". This rule derivation procedure is illustrated in Figure 4. (for analysis of an electric toothbrush product) As a first step, we create the 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 4. In the first rule, the rule states that the functional requirements of "convert rotational mechanical energy to translational mechanical energy" and "transfer translational mechanical energy" are addressed in the design by the use of the component "link". Similarly, the last rule shown in Figure 4 indicates that the function "transfer RME" in the function structure is solved by a "driveshaft" and a "rotational coupler" in the actual design.
Note that the rules in Figure 4 are not simply one-to-one matches of functions to components. The open-endedness of the grammar formulation allows us not to be limited by one-to-one mappings between functions and components. Accordingly, the rules may map a single function to multiple components, or multiple functions to a single component, or multiple functions to multiple components. Following this procedure, we currently have defined 170 rules derived from 17 products.

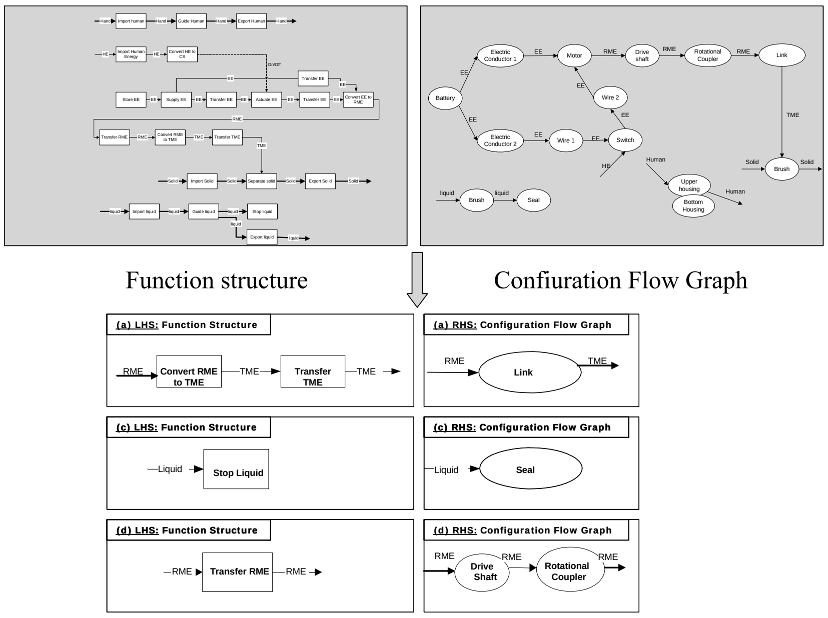

Figure 4. llustration of rule derivation (only three rules) from empirical analysis of products

The grammar provides an effective method to generate design configurations through the execution of rules that create feasible solutions to the design problem. In this research, feasible solutions are represented by a set of CFG's. Our synthesis methodology is to perform a graph transformation of the initial function structure into a configuration flow graph. 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 compromised of a left hand side (LHS) and right hand side (RHS) as illustrated in Figure 4. The left-hand side contains the state that must be recognized in the function structure and the right-hand side depicts how the design is transformed to a new configuration by the addition of new component(s). The basic generation process for a set of rules is to first recognize which rules have left-hand sides that match the current state, then choose one of these rules, and finally to apply the rule as a step towards constructing an updated configuration. This cycle is repeated until no further rules are recognized, indicating that all functional requirements given by the function structure are addressed and that the configuration synthesis is complete.

5. Illustrative Example: Design of a Bottle Capping Device

As an initial implementation of our design method, we designed a "bottle capping device". This problem was insipired by a past ASME Student Design Competition. The selected design case required the development of a machine that would register a bottle to a capping station, cap it, and allow somebody to retrieve the capped bottle from the device.

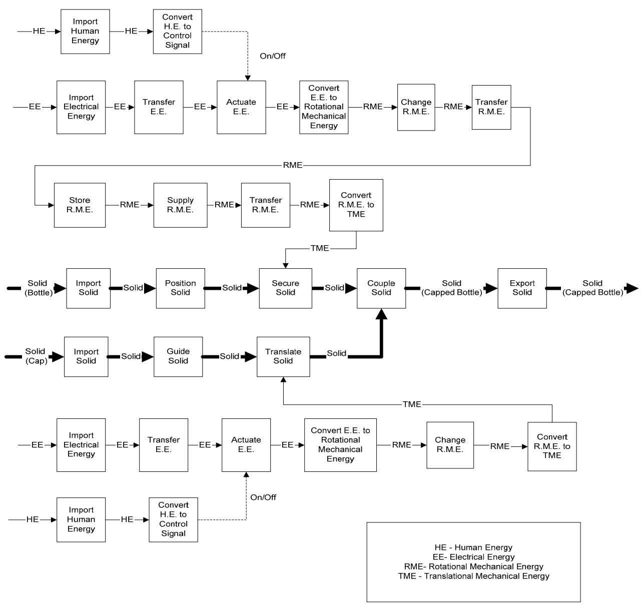

Figure 5. The function structure for the bottle capping device to be designed

The design process starts with the specification of the function structure of the product to be designed (shown in Figure 5). This is accomplished through a graphical user interface that allows the designer to quickly draft a function structure. Figure 6.a shows the user interface with the initial function structure drawn (partially) for the bottle capping device. The "input external" and "output external" nodes in figure represent dummy nodes that ensure the input and output flows are not left dangling. Dangling arcs are not easily displayed in the implemented GUI and are rarely considered as valid connections in many graph theory applications. The program initiates the design generation process in which sub-functions are replaced with components through application of rules. At the beginning none of the sub-functions are mapped to components, thus the CFG starts as an empty graph. As the rules are applied, the CFG is incrementally updated by the replacement of sub-functions with new components as described by the selected rules. Figure 6.b shows the state of the design after the application of "spring" and "driveshaft" rules. 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.

In our current implementation, the rules that are actually applied are selected automatically as we exhaustively search the design space for all possible configurations. The result of this search is a set of complete design configurations (CFG's), which are then presented to the designer. Samples of the resulting CFG's for the bottle capping device design are shown in detail at http://www.me.utexas.edu/~adl/cfg_grammar.htm.

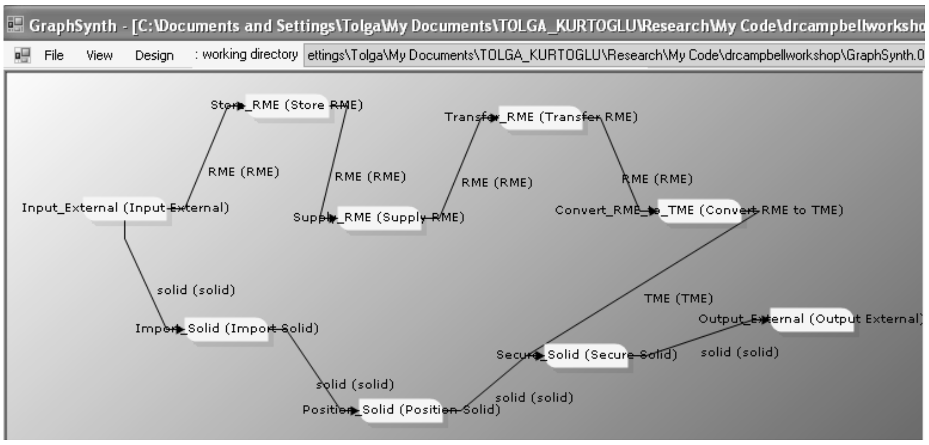
GraphSynth - [C:\Documents and Settings\Tolga\My Documents\TOLGA_KURTOGLU\Research\My Code\drcampbellworksho
File
View
Design
: working directory
ettings\Tolga\My Documents\TOLGA_KURTOGLU\Research\My Code\drcampbellworkshop\GraphSynth.0
Store_RME (Store RME)
Transfer_RME (Transfer RME)
RME (RME)
RME (RME)
RME (RME)
RME (RME)
Input_External (Input External)
Supply_RME (Supply RME)
Convert_RME_to_TME (Convert RME to TME)
solid (solid)
TME (TME)
Output_External (Output External)
Import_Solid (Import Solid)
Secure_Solid (Secure Solid)
solid (solid)
solid (solid)
solid (solid)
Position_Solid (Position Solid)

Figure 6a. Graphical user interface of the developed automated concept generation software. The designer sketches the function structure of the system that is to be designed

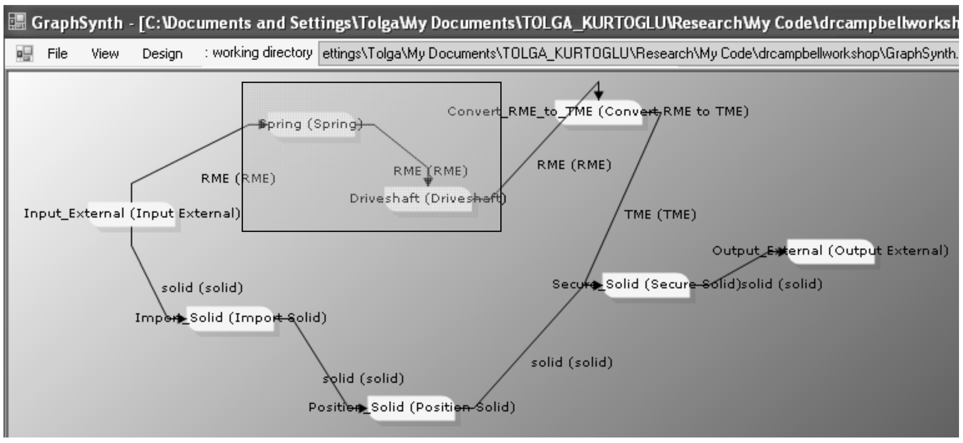
GraphSynth - [C:\Documents and Settings\Tolga\My Documents\TOLGA_KURTOGLU\Research\My Code\drcampbellworksh
File
View
Design
: working directory
ettings\Tolga\My Documents\TOLGA_KURTOGLU\Research\My Code\drcampbellworkshop\GraphSynth.
Spring (Spring)
Convert_RME_to_TME (Convert RME to TME)
RME (RME)
RME (RME)
RME (RME)
Driveshaft (Driveshaft)
Input_External (Input External)
TME (TME)
Output_External (Output External)
Secure_Solid (Secure Solid)
solid (solid)
solid (solid)
Import_Solid (Import Solid)
solid (solid)
solid (solid)
Position_Solid (Position Solid)

Figure 6b. A screenshot of the user interface at a partially completed design state

In creating these design configurations, the program successfully integrates component concepts from different products into one complete concept variant. 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 past component solutions which have been successfully used in the design of existing products.

6. 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 creating new concept variants. The design knowledge is captured through an empirical study that involves systematic dissection of products and the construction of an online design repository. The extracted design knowledge is then incorporated into a computational synthesis framework that utilizes graph-based design representations and associated graph grammar and search algorithms for the generation of new designs. 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 as accomplished by the grammar. Secondly, the psychological bias is removed that limits designers to specific engineering domains. Finally, the likelihood of design success is increased by the use of the presented method, simply because the methodology itself is based on leveraging successful realizations of the past conceptual solutions.

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 evaluation methods, which will allow a designer to sort or rank generated design concepts. Although these evaluation metrics are yet to embrace non-technical aspects of customer needs such as aesthetics and ergonomics, they will provide a framework for a first-pass computational evaluation of the feasible concepts based on a variety of higher-level customer needs including compactness, reliability, weight, manufacturability, and cost.

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.,"A Computational Technique for Concept Generation". Proceedings of ASME 2005 International Design Engineering Technical Conference, IDETC'05, Long Beach, CA, 2005.

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

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

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

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

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

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

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

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

Tolga Kurtoglu, Graduate Research Assistant, Ph. D. Candidate
The University of Texas at Austin, Department of Mechanical Engineering
One University Station C2200, Austin, TX, 78712, USA
Tel.: +1 - (512) - 4717347
Email: tolga@mail.utexas.edu
URL: http://www.me.utexas.edu/~adl/tolga.htm