Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2010), *24*, 49–62.
Copyright © Cambridge University Press, 2010. 0890-0604/10 $25.00
doi:10.1017/S0890060409990163

Automating the conceptual design process: "From black box to component selection"

TOLGA KURTOGLU,[1] ALBERT SWANTNER,[2] AND MATTHEW I. CAMPBELL[2]
[1]Mission Critical Technologies, NASA Ames Research Center, Intelligent Systems Division, Moffett Field, California, USA
[2]Automated Design Laboratory, Department of Mechanical Engineering, University of Texas at Austin, Austin, Texas, USA

(Received October 15, 2008; Accepted July 3, 2009)

Abstract

Conceptual design is a vital part of the design process during which designers first envision new ideas and then synthesize them into physical configurations that meet certain design specifications. In this research, a suite of computational tools is developed that assists the designers in performing this nontrivial task of navigating the design space for creating conceptual design solutions. The methodology is based on automating the function-based synthesis paradigm by combining various computational methods. Accordingly, three nested search algorithms are developed and integrated to capture different design decisions at various stages of conceptual design. The implemented system provides a method for automatically generating novel alternative solutions to real design problems. The application of the approach to the design of an electromechanical device shows the method's range of capabilities and how it serves as a comparison to human conceptual design generation and as a tool suite to complement the skills of a designer.

Keywords: Automated Design; Concept Generation; Functional Design; Graph Grammars

1. INTRODUCTION

Conceptual design plays the central role in ensuring design quality and a high level of innovation. It is in this phase that the architecture of the final design is established, the technologies are chosen to fulfill the customer needs, and the bulk of the cost for a product is committed. Because of these characteristics, conceptual design is often considered the most important phase of the product development cycle.

Yet, the conceptual design process has seen few attempts at automation. The concept of "automating design" has often been leveraged in later stages of the design process where a to-be-designed artifact accrues numerous parameters but lacks specific dimensions. Automated methods such as optimization provide a useful framework for managing and determining details of the final designed artifact. These methods make the design process less tedious and time-consuming, and they are used in a wide variety of industries to support or optimize current design efforts. However, one of the pervasive bottlenecks in design is the lack of continuity between computational design tools and conceptual design methods.

Reprint requests: Tolga Kurtoglu, Mission Critical Technologies, NASA Ames Research Center, Intelligent Systems Division, Moffett Field, CA 94035, USA. E-mail: tolga.kurtoglu@nasa.gov

The difficulty may hinge on the very nature of conceptual design, which does not lend itself easily to automation. The conceptual design process begins with the specification of the product to be designed and involves the continual cycle of concept generation and evaluation until a design opportunity is transformed into an embodied solution that satisfies a set of design requirements.

This systematic view of conceptual design starts with the formulation of the overall function of the product to be designed. This high level product function is then decomposed recursively into lower level functions—a process that produces a function structure, which is a representation that defines function as transformation between energy, material, and information (Pahl & Beitz, 1996). The function structure is then used to generate solutions to each of the product subfunctions. Here, the designer seeks solutions (a component or a set of components that perform a particular function). Next, solutions to the subfunctions are synthesized together to arrive at the final architecture or configuration of a product. Finally, the design is embodied by the selection of designed components (Suh, 1990; Ullman, 1995; Ulrich & Eppinger, 1995; Pahl & Beitz, 1996; Otto & Wood, 2001). Using this systematic view of conceptual design, a broad number of concepts can be generated by making decisions about the decomposition of the overall product function and the selection and integration of different design solutions to elemental subfunctions.

In this research, we automate the aforementioned conceptual design process starting from a black box[1] level product specification to the physical embodiment of design components. Accordingly, we develop a suite of automation tools that combine and formalize the systematic view of the conceptual design process (Suh, 1990; Ullman, 1995; Ulrich & Eppinger, 1995; Pahl & Beitz, 1996; Otto & Wood, 2001). The implemented system consists of three nested search algorithms that can capture different design decisions at various stages of conceptual design and serves as a comparison to human conceptual design generation.

The remainder of this paper is broken up into six sections. The next section talks about the different approaches and techniques developed to automate the conceptual design process. An overview of the proposed method and the three search algorithms are described in Section 3. Section 4 presents the implementation of the method on a case study in which an electromechanical device is synthesized. Various results obtained from this example are analyzed in Section 5. Finally, the conclusions are presented in Section 6.

2. RELATED WORK

Various researchers have employed different methods to computationally support the conceptual phase of design. These methods include computer techniques such as constraint programming (Kota & Chiou, 1992; Subramanian & Wang, 1995), qualitative symbolic algebra (Williams, 1990), expert systems (Mittal et al., 1985), or case-based reasoning (Navinchandra et al., 1991; Bhatta et al., 1994; Qian & Gero, 1996). Among these, one of the most historically significant is the expert system formulation described in the PRIDE system established by Mittal et al. (1985), which was specifically developed for creating paper roller systems. A subset of expert systems, case-based reasoning techniques apply past knowledge stored in a computational database toward solving problems in similar contexts. Examples include Qian and Gero (1995), who presented a system called FBS, which uses relations among function, behavior, and structure to retrieve design information to conduct analogy-based design. Similarly, the structure–behavior–function modeling scheme (Bhatta et al., 1994) and its computational application KRITIK is a system relying on a design-case memory to conduct computational synthesis.

Apart from expert system formulations, typical examples of computational synthesis applications start with a set of fundamental building blocks and some composition rules that govern the combination of these building blocks into complete design solutions. Hundal (1990) designed a program for automated conceptual design that associates a database of solutions for each function in a function database. Ward and Seering (1989) developed a mechanical design "compiler" to support catalog-based design. Bracewell and Sharpe (1996) developed "Schemebuilder," a software tool using bond graph methodology to support the functional design of dynamic systems with different energy domains. Chakrabarti and Bligh (1996) model the design problem as a set of input–output transformations, where structural solutions to each of the instantaneous transformation are found, and infeasible solutions are filtered according to a set of temporal reasoning rules. Bryant et al. (2005) developed a concept generation technique that utilizes a function–component matrix and a filter matrix to generate a morphological matrix of solutions during conceptual design. The A-Design research (Campbell et al., 2000) is an agent-based system that synthesizes components based on the physical interactions between them.

Function structure research, in contrast, has found its way into a number of educational texts since the presentation provided by Pahl and Beitz (1996). Furthermore, research building upon the concept of function structures has flourished in the past 15 years. Numerous publications have extended the application of function structures and formalized the use of such structures (Otto & Wood, 1997; Umeda & Tomiyama, 1997; Kirshman & Fadel, 1998; Kitamura & Mizoguchi, 1999; Stone & Wood, 1999; Szykman et al., 1999; Hirtz et al., 2002). Computational approaches have also been explored that further expand the value of function structures (Wang & Yan, 2002). One of the interesting implementations of automating the function-based design is the work of Sridharan and Campbell (2004), which uses graph grammars.

Graph grammars are composed of rules for manipulating nodes and arcs within a graph. The rules create a formal language for generating and updating complex designs from an initial graph-based specification. The development of the rules encapsulate a set of valid operations that can occur in the development of a design. Through the application of each grammar rule the design is transformed into a new state, incrementally evolving toward a desired solution. The rules are established prior to the design process and capture a certain type of design knowledge that is inherent to the problem. The knowledge captured in the rules offers the option of exploring different design decisions and thus different design alternatives.

Using this formalism, Sridharan and Campbell (2004) defined a set of 69 grammar rules that were developed to guide the design process from an initial functional goal to a detailed function structure. Elsewhere, graph grammars are widely used in various engineering applications. Agarwal and Cagan's (1998) coffee maker grammar was one of the first examples of using grammars for product design. Their grammar described a language that generated a large class of coffee makers. Shea et al. (1997) presented a parametric shape grammar for the design of truss structures that uses recursive annealing techniques for topology optimization. Other engineering applications include Brown and Cagan (1997), who presented a lathe grammar, Schmidt and Cagan's (1995) grammar for machine design, Starling and Shea's (2003)

[1] The black box representation defines the energy, material, and signal flows entering and leaving an artifact.

grammars for mechanical clocks, and gear trains (Starling & Shea, 2005).

Although these methods are primarily concerned with generation aspects of conceptual design, there are various techniques developed to automate the selection of components for an already generated design configuration. These techniques include using genetic algorithms, simulated annealing, and integer programming. Weilinga and Schreiber (1997) classify the component selection problem as category one within their work, where the set of components as well their assembly is fixed. Carlson (1996) uses a genetic algorithm for component selection given a user-defined system layout, a database of components and a set of design specifications. Carlson et al. (1998) apply a genetic algorithm for solving the problem of catalog design. They create an initial set of components types followed by component selection from the component database. Dallaali and Premaratne (2004) apply genetic elitism and double string coding to solve optical component selection problems.

In summary, our background research shows that a number of attempts have been made to automate various key elements of the design process such as the creation of function structures, configuration design, and component selection. However, most of these methods have been developed for specific applications. These methods have also been restricted to a specific phase or task of conceptual design. The method presented here is a generalized technique that follows the grammar formalism and integrates it with fundamentals of the function-based synthesis paradigm to automate the design decision making that governs the entire concept generation process starting at a black box level product specification and finalized by the selection of components that physically embody the design. The details of this design automation approach are explained next.

3. RESEARCH APPROACH

This research aims to automate the systematic design process (presented in Pahl and Beitz, 1996). Accordingly, it extends the previous automated design research by developing a suite of computational design tools that transform a high-level, functional description of a nonexistent product into a set of embodied concept variants by following the process shown in Figure 1.

By automating this process, a design is changed from an abstract set of customer needs to an embodied conceptual configuration. The customer need analysis and the formulation of the initial "black box" steps are performed by the designer. The computational design synthesis is initiated at the level of a black box. The first design tool (the function structure grammar) converts the black box of a device to be designed into a set of detailed function structures by using functional decomposition rules. Based on this functional input, the second design tool (the configuration grammar) synthesizes individual or sets of components into a set of conceptual design configurations. Finally, the third design tool (the tree search algorithm for component selection) instantiates specific components in a design configuration guided by specific design constraints and objectives. In the following paragraphs, each of the three design tools and their specific search algorithms are explained in detail.

3.1. Research effort I: Function structure grammar

A common technique in phrasing the problem as a black box is useful in engineering design to clarify the goals of the project (Pahl & Beitz, 1996; Otto & Wood, 2001). By removing all unnecessary information, the black box defines only the flows entering and leaving the product. The black box is often labeled with a primary function, which is typically a verb–noun pair. The first automated design tool (Sridharan & Campbell, 2004) acts on this black box input to automatically create the necessary functions for translating the input flows into the output flows. To accomplish this, a series of 52 rules have been created based on the data of 30 black boxes and their corresponding function structures. These rules are created to capture common function chains used in a variety of artifacts. For example, an automobile jack performs a function that requires energy from the human; but instead of being applied directly, it is first converted to pneumatic energy and then converted to mechanical energy. This series of functions is also seen in a toy NERF gun among other applications. This inference of common modules can help make better function structures, as it is not always easy for a designer to make a connection between such different products. To ensure consistency for the representation of the function structures and the consequent rule development, we have adopted the functional basis taxonomy (Hirtz et al., 2002). The functional basis is a set of function and flow terms that combine to form a subfunction description (in verb–object format) and that are utilized during the generation of a black box and functional model to encapsulate the actual or desired functionality of a product.

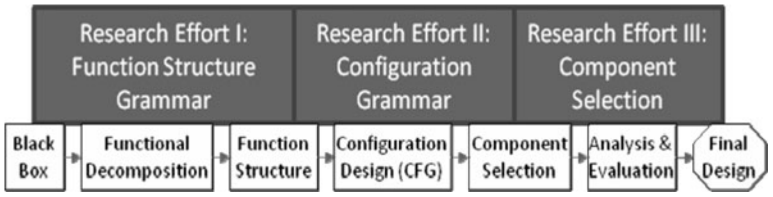
Research Effort I: Function Structure Grammar
Research Effort II: Configuration Grammar
Research Effort III: Component Selection
Black Box
Functional Decomposition
Function Structure
Configuration Design (CFG)
Component Selection
Analysis & Evaluation
Final Design

Fig. 1. Three nested search algorithms automate the conceptual design process.

Starting with the black box, the grammar rules enumerate all possible valid function structures. This can be viewed as a search tree, where the black box is the seed or start node, and the rules provide the transition operators that lead to the many leaves of the tree. It is interesting to note that the black box is a detailed start state for the tree, and the existence (or absence) of input and output flows limits the size of the search tree significantly. This is shown clearly in the results of this paper. In the future, the rules may be revisited so as to create or modify specified input and output flows of the black box, but it is unclear how these will be regulated.

The original function structure grammar (Sridharan & Campbell, 2004) was modeled in the same way grammar rules are modeled, as independent if–then statements. This is often shown graphically with left-hand and right-hand sides (Fig. 2a). In terms of implementation, this work was done *ad hoc*, resulting in a large and unorganized set of java files. Recently, these rules have been rewritten in a new graphical environment known as GraphSynth (Campbell, 2007) that allows one to graphically create the rules and manages the resulting data as a series of portable xml files (see Fig. 2b). Note that in Figure 2a, the gray and black circles, which are referred to as "active centers," serve as markers to ease the implementation. This concept of active centers can be found in nature in the chemical polymerization process. In addition to polymerization, unsaturated molecules add onto a growing polymer chain one at a time. The last molecule on the chain is the *active center* upon which the unsaturated molecules can attach. Similarly, during the creation of a function structure, there are many active centers where incoming flows and functions can attach themselves. These active centers are the points where grammar rules can be applied and where new functions and/or flows are added if certain criteria are met at a specific open connection. In recreating these rules (Fig. 2b), the concept is maintained but no longer required as GraphSynth includes a more general and powerful subgraph recognition procedure than using active centers. In either case, the rule provides guidance in developing the connecting flows to a "remove" function. This rule captures the principle that whenever we cut or grind a solid, we need to supply some mechanical energy, and this results in two or more pieces of the solid. It should also be noted that this rule cannot be applied again because the active center necessary for rule recognition has been eliminated. Care must be taken to define rules that prevent multiple applications of the same rule because this would cause an endless loop where no functionality is being added, only duplicates of existing functionality.

The rule shown in Figure 3a captures another common principle. When mechanical energy is supplied, the effort of the energy, torque, can be amplified and this is represented by the function "Change ME." This function can be satisfied by using a gear train. This rule looks for a flow of type mechanical energy that is open at the tail and is pointing to the function, "Remove Solid." If applied, this rule adds the function "Change ME" to the tail and adds another flow open at its tail to the back of the "Change ME" function. Figure 3b shows a rule where an electric energy flow that is pointing from "Import" is recognized and the functions "Transmit EE" and "Actuate EE" are added to it. This rule is observed in many products that use electrical energy, as electrical energy is always transmitted and actuated before being

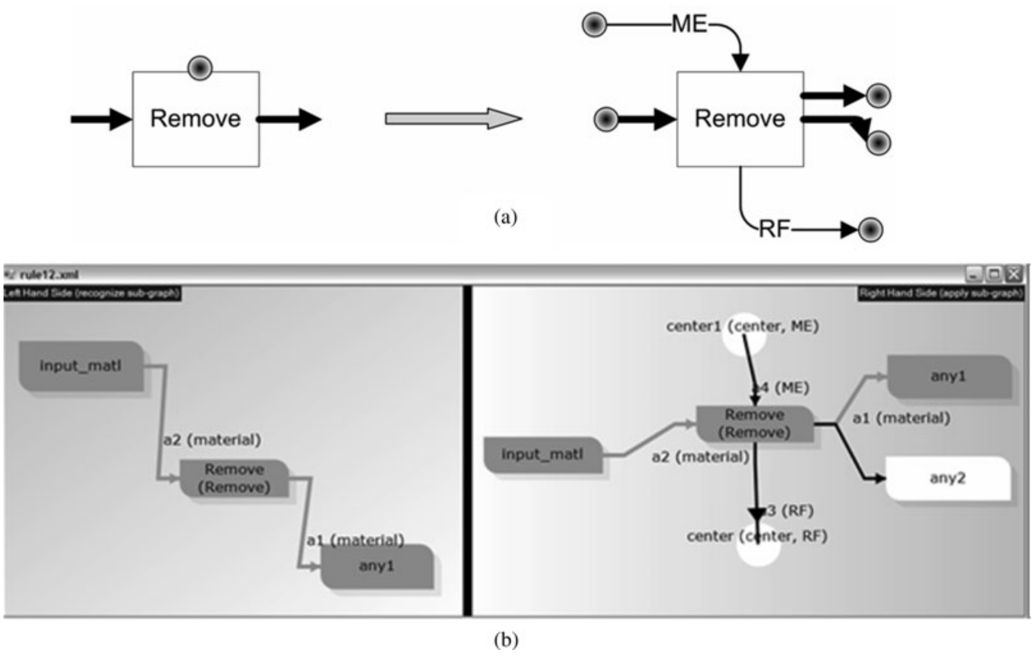

Fig. 2. An example rule shows the use and propagation of active centers (a) as shown in the original publication (Sridharan & Campbell, 2005) and (b) as recreated in GraphSynth.

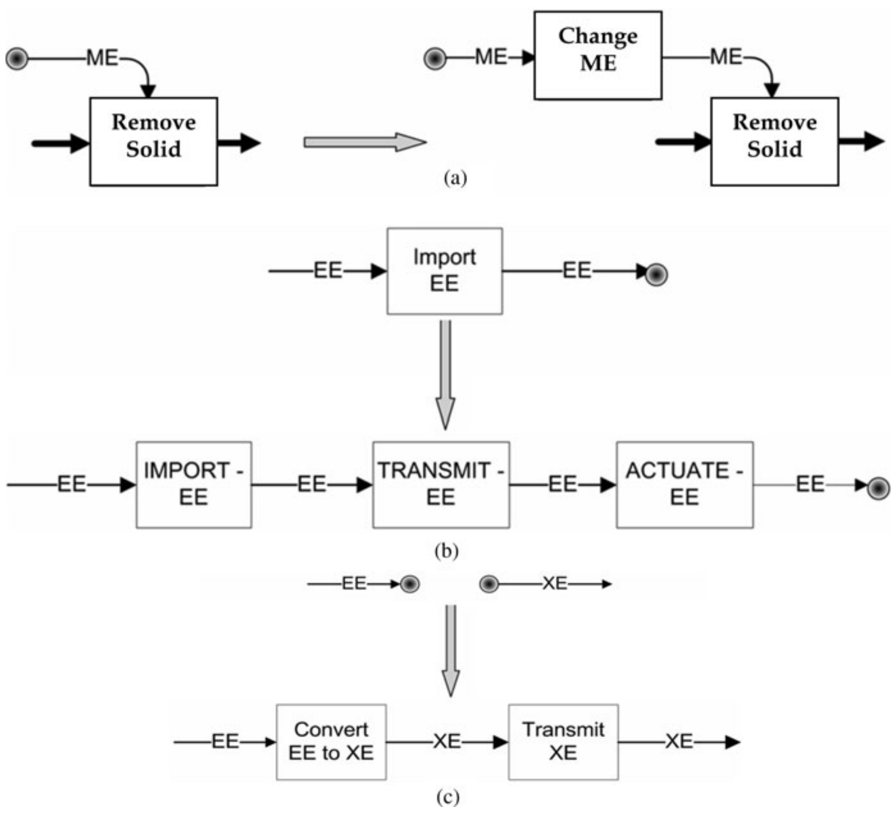

Fig. 3. Three additional examples of function structure grammar rules (Sridharan & Campbell, 2004).

converted to the required form. The final rule shown in Figure 3c is a termination rule where active centers are destroyed. Whenever two flows are recognized such that we have an open electrical energy flow and energy of any other kind (represented as XE) that needs to be supplied, we convert the electrical energy to the required form and transmit it. Termination rules are vital in obtaining a valid function structure. A valid function structure ensures that all of the flows that go into the system are utilized and then exported in some form. If the termination rules were not called, then energy, material, or signal chains could dead end and it would be impossible to continue with the other steps of the design process. Figure 3 is based on the work of Sridharan and Campbell (2004).

3.2. Research effort II: Configuration design grammar

For decisions at the conceptual phase of design, the interconnectivity of design elements is more important than parametric details (Kurtoglu, 2007). In such conceptual design problems, it becomes essential to determine an optimal configuration of components prior to tuning individual component parameters. Creating such configurations is the objective of the second design tool: the configuration design grammar.

The starting point for the configuration design grammar is a function structure. To maintain consistency between the two research efforts, the secondary level of the functional basis taxonomy (Hirtz et al., 2002) is used as the representational language for expressing this input function structure. The synthesis process is aimed to perform a graph transformation of a function structure into a set of configuration-based graphs called the Configuration Flow Graphs (Kurtoglu et al., 2005). In a configuration flow graph (CFG), nodes represent design components and arcs represent energy, material, or signal flows between them. The graph is also similar to an exploded view in that components are shown connected to one another through arcs or assembly paths. Using a CFG, designers can capture components that are present in a design, their connectivity, and physical interfaces between a design's components. Figure 4 shows an example of a CFG for a disposable camera.

The grammar rules for the configuration design are defined through a knowledge acquisition process that is based on the dissection of existing electromechanical devices. Accordingly, for each device that is dissected, a function structure and a configuration flow graph are generated. Then, the mapping between the two graphs is captured where each mapping represents a potential grammar rule (Kurtoglu & Campbell, 2005). Some of the rules derived from this analysis are shown in Figure 5. Note that the rules in Figure 5 are not simply one-to-one matches of functions to components. The open-endedness of the grammar formulation allows us to tend to assign single components to single functions.

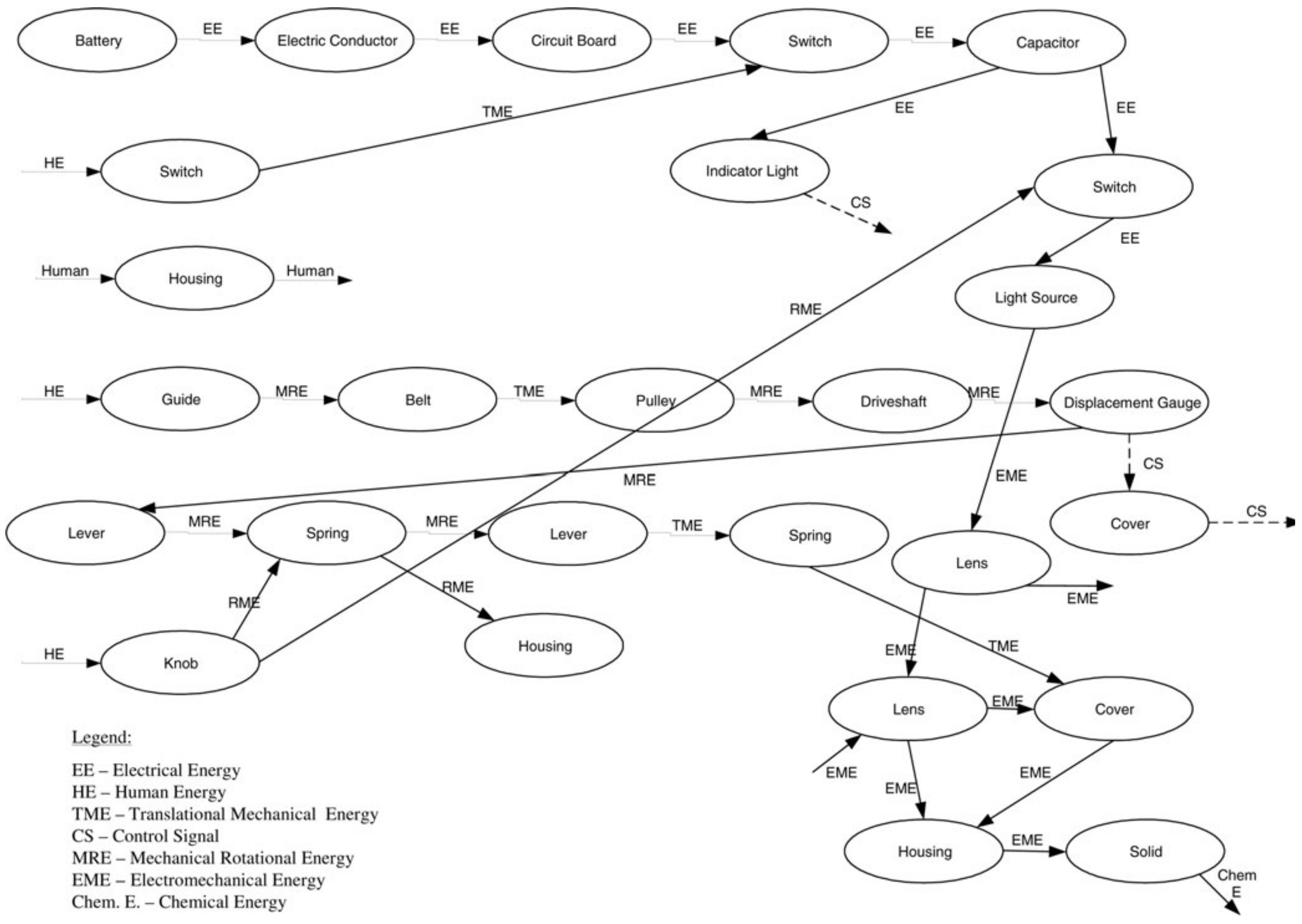

Fig. 4. The CFG of a disposable camera.

Instead, through multiple node recognition and application the grammar provides a more generic approach capable of inserting multiple components for a single function, a single component for multiple functions, as is the case in function sharing, or multiple components for multiple functions.

In reality, each rule represents a design decision that shows how a functional requirement was transformed into an embodied solution in an actual design. Currently, the rule database contains 170 grammar rules derived from the dissection of 23 electromechanical products. These rules and additional information about these products are stored in a Web-based Design Repository (Design Engineering Lab, Missouri University of Science and Technology; http://function.basiceng.umr.edu/repository) that is managed at the Missouri University of Science and Technology. The grammar provides an effective method for automatically generating design configurations through a search-based execution of rules. The computational synthesis approach performs a graph transformation of the initial function structure of the to-be-designed product into a set of configuration flow graphs. Each execution of a rule adds more components to the design configuration, which incrementally builds to a final concept. At the end, the computational search process returns different concepts with potentially varying degrees of complexity as candidate configurations to the same functional specifications.

In detail, the transformation from the function structure to a CFG is part of a recognize–choose–apply cycle. The recognize step identifies all possible locations in the function structure where a grammar rule can be applied. These locations define a set of possible graph transformations that can be applied at that design stage. This step is followed by choosing one of the valid grammar rule options. In the final apply step, the CFG is updated per the instructions provided by the selected rule. This process is repeated until there are no more rules that can be applied.

The final configurations obtained at the end of this generation process depend on the selection of the rules applied. To fully automate the generation process, this selection is made by the computer. The basis and the guidelines to select the rules are embedded in the search algorithm. In the current implementation, each applicable grammar rule is systematically selected by the computer as the configuration space is traversed using a breadth-first search (BFS) approach.

At the end, the search process generates a variety of configurations that are developed from a functional description of

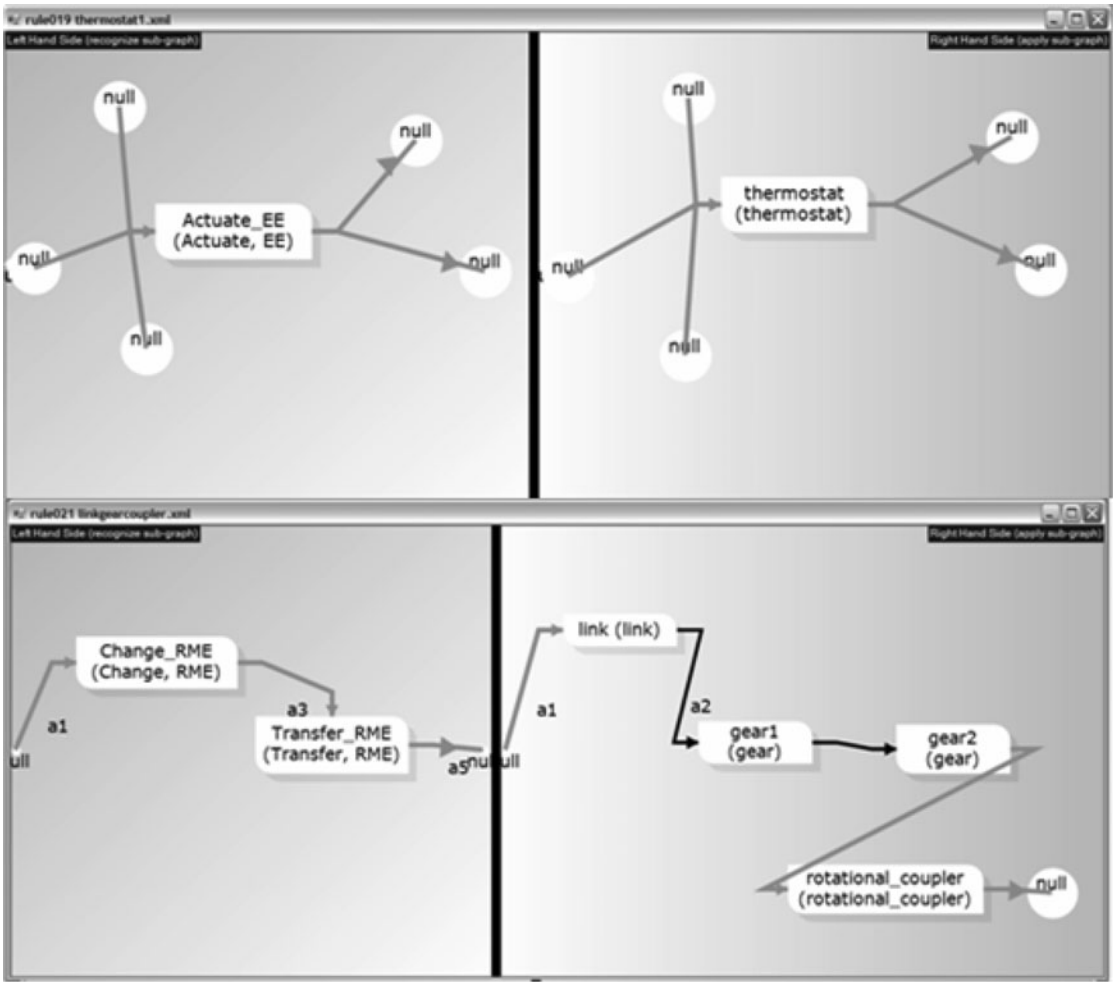
rule019 thermostat1.xml
Left Hand Side (recognize sub-graph)
Right Hand Side (apply sub-graph)
null
Actuate_EE (Actuate, EE)
thermostat (thermostat)
rule021 linkgearcoupler.xml
Change_RME (Change, RME)
Transfer_RME (Transfer, RME)
a1
a2
a3
a5
link (link)
gear1 (gear)
gear2 (gear)
rotational_coupler (rotational_coupler)

Fig. 5. Two grammar rules of the configuration grammar in GraphSynth. The left-hand side of the rules captures the functional requirement and the right-hand side depicts how the functional requirement can be addressed by the use of specific component concepts.

a product by synthesizing component solutions together that have been successfully used in the design of past products.

3.3. Research effort III: Tree search algorithm for component selection

The objective of the third design tool is to determine the optimal choice of components for a specified CFG (Tamhankar & Campbell 2007). To accomplish this, the components are chosen from a database, which contains a compilation of real component information for each component abstraction (such as electric motor, bearing, shaft, and gear) that can be present in a CFG. This data for each component, or artifact, has been collected from an online repository created and maintained by the Missouri University of Science and Technology (2007) as well as online product catalogs (2007).

The approach is an iterative process that replaces each component in the CFG with an artifact that is stored in the database. Each component in a CFG represents a different level of the search tree, and the number of options at each level is equivalent to the number of artifacts in the database for that particular component. Each node further down the tree replaces a generalized CFG component (e.g., gear) with an artifact (e.g., steel plain bore 14.5° pressure angle spur gear with 24 teeth, a pitch of 32, and a face width of 3/16 in.). The branching factor[2] thus depends on the number of choices for a component, whereas the number of levels in the tree is the number of selections to be made as determined by the CFG. Currently there are, on average, six artifacts per component (there is only one electrical cord but 20 gears). As the search process unfolds, more components are instantiated by replacing abstract components in the CFG with actual artifacts from the database.

At this stage in the engineering design process it is possible to include some evaluation of the design decisions. In this research, an objective function was constructed that combined various customer needs (such as minimize cost and maximize power) with compatibility metrics for adjacent components (e.g., how different is the shaft diameter from the

[2] In computing and tree data structures the branching factor is the average number of children from each node in the tree.

mating gear's bore diameter). These customer need satisfaction and compatibility equations play a key role in the final component selections. There is an element of subjectivity and dependence on the designer for the assignment of weights and penalty that cannot be eliminated.

The space of solutions found in the tree is searched using a uniform cost search algorithm. The search begins at the root node (a complete CFG) and the traversal of the tree is employed by instantiating one component at a time. During the search, it is possible to evaluate the design decisions governing the instantiation of components. Accordingly, an objective function is constructed that combines criteria measuring how well various customer needs (such as minimize cost and maximize power) are met with compatibility metrics for neighboring components (e.g., how different is the shaft diameter from the mating gear's bore diameter). At each node of the tree, the node expansion is performed after calculating transition costs based on this objective function formulation. These transition costs are additive in nature, and at each step only the child with the minimum transition cost is generated. This search process continues until the CFG is fully instantiated and an optimum solution is reached.

4. CASE STUDY: DESIGN OF A COFFEE GRINDER

The proposed methodology is demonstrated in this section by solving a test problem: the design of a coffee grinder. In this problem, we start off by creating a simple black box and illustrate how the computer can generate associated function structures and configuration flow graphs.

Figure 6 shows a black box, the primary function of which is "Change Solid." The primary function, along with the input and output flows, of the black box are determined by the designer. These decisions govern the energy domains, materials, and signals, which the product would need to use. For the selected problem, it is envisioned that the designed artifact would primarily utilize human and electrical energy for the input energies and that a user would actuate the device operation. These design decisions are captured by the specification of human material and human and electrical energy input flows as shown in Figure 6. Similar decisions are made for the output flows. Accordingly, it is specified that the machine would use mechanical energy to perform the "change" function and that the human material would be returned. The specification of input and output flows poses constraints to the design problem and keeps the artifact choices in certain domains. By specifying electrical energy as input and mechanical energy as output, we limit the functions that can be called and the consequent variety of components that can be selected. The specification of the primary function and the input and output flows ensures that the customer needs are captured before the design process starts and that the computer will not end up with solutions that the user did not intend or is not interested in.

The complete search process is run using the GraphSynth environment (Campbell, 2007). The process starts with the user drawing the black box of the to-be-designed artifact. The function structures and CFGs are then created automatically using their respective sets of grammar rules. The results for the selection of the components are yet to be implemented for the presented design example as we continue working on formulating an objective function to evaluate the performance of the coffee grinder.

5. RESULTS AND DISCUSSION

For the implementation of this case study, the algorithms are run on a Windows PC with 3.25 GB of RAM and a 2.2-GHz processor. After the input is specified as shown in Figure 6, the computational synthesis of the coffee grinder begins, initially with the creation of potential function structures for the design.

The function structure grammar makes use of three rule sets. The first rule set encompasses initiation rules and inserts active centers to the graph for each of the input and output flows as described in Sridharan and Campbell (2004). The second rule set of the grammar, called the propagation rule set, generates all of the functions in the functional model. This model is built utilizing two directions starting from both the input and the output (left and right as seen in Fig. 6) flows. After the propagation rules are executed some functions may not be fully connected. The final termination rule set ensures that these connections are not left dangling and that the generated function structure is complete.

For the coffee grinder problem, the function structure grammar ran for 4 h and created two unique function structures.

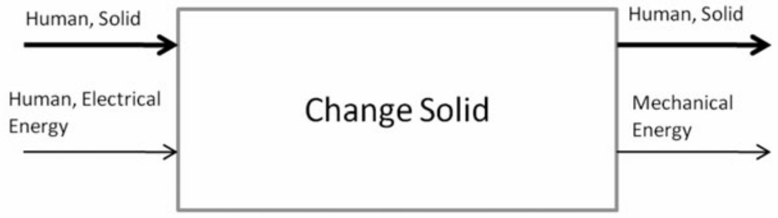

Fig. 6. The black box for a coffee grinder. Note the flows and primary function are simplified so that it is understandable to the computer.

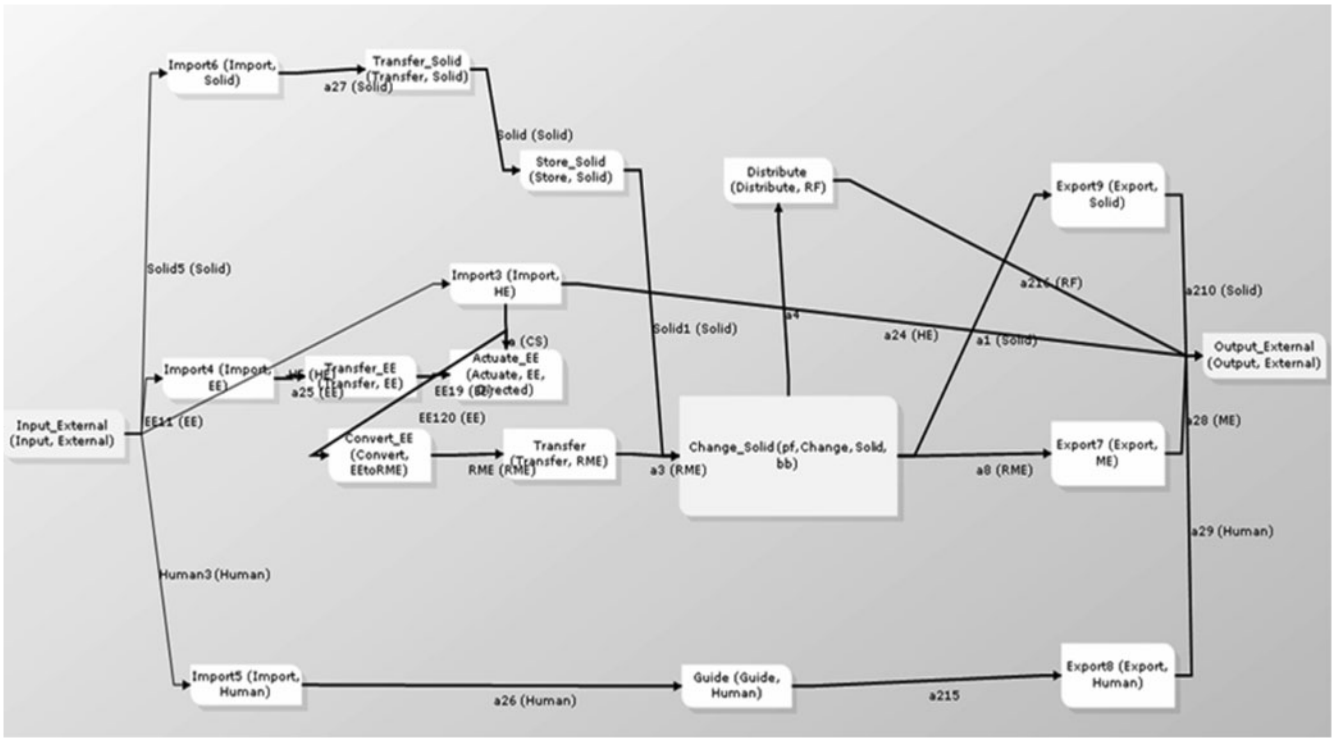

Fig. 7. The second candidate function structure created automatically from the black box shown in Figure 6.

These are shown in Figure 7 and Figure 8. The generated function structures are similar in nature and only differ in their use of mechanical energy. One of them uses translational mechanical energy to perform the grinding operation while the other one uses rotational mechanical energy.

These two unique function structures are then posed as inputs into the second design tool, which takes the function structure as a starting point and generates conceptual configurations. Specifically, the computer employs a modified BFS algorithm that includes a filtering mechanism that removes duplications of previously visited nodes from the search space. This filtering is based on a property in graph grammar theory known as confluence, which means that the order in which a subset of rules is invoked does not affect the final outcome. To better illustrate this concept, let us consider three rule choices, A, B, and C, that must be called to complete a candidate design. After making the first choice it appears as if one may have three unique solutions, A__, B__, and C__. After the second choice, the number of possible unique solutions stays at three, AB__, AC__, BC__, but these solutions are interpreted by the computer using six possible branches because the computer cannot differentiate between the designs AB and BA. Finally, it is not until the last iteration that one can conclude that there is only one unique solution. This is shown more clearly in the bar graph of Figure 9, where the number of candidates is shown on the *y*-axis and the number of rules called is depicted on the *x*-axis. As the rules are invoked the number of candidates quickly expands only to be followed by a decrease. The reason for this is similar to the simple example given above where the computer cannot tell that there are three identical copies of the design in the search tree until later in the process. As the plot shows, the number of unique candidates needs to reach a critical point after which the filtering takes effect. This filtering cuts the amount of time that the process took by a factor of 20 and greatly reduces the number of candidates at each level.

The fourth set of grammar rules (the configuration grammar) ran for about 5 days and generated 1536 unique solutions. This many unique solutions is still an overwhelming number for a human designer to consider, but the natural reduction from over 50,000 is interesting. Two of the unique solutions are shown in Figure 10 and Figure 11. The CFG in Figure 10 was derived from the function structure in Figure 7 and the CFG in Figure 11 was derived from the function structure in Figure 8. These CFGs are different from each other in many ways, but strictly speaking, for a CFG to be unique, only one component needs to be different. For example, if the "transfer mechanical energy" function is accomplished by a shaft in one CFG and by a conveyer in another, those two CFGs are considered to be unique. In the two CFGs shown here, there are many differences. The first of the CFGs takes the mechanical energy from the motor and goes straight to the shaft to the blade. In the second CFG, this mechanical energy goes from the motor to a gripper then to a support, a sprocket, and then the blade. This second chain of energy may spark an interesting idea by the designer on a way to get rid of the costly shaft, or it may be deemed too complicated for this application and disregarded. The purpose

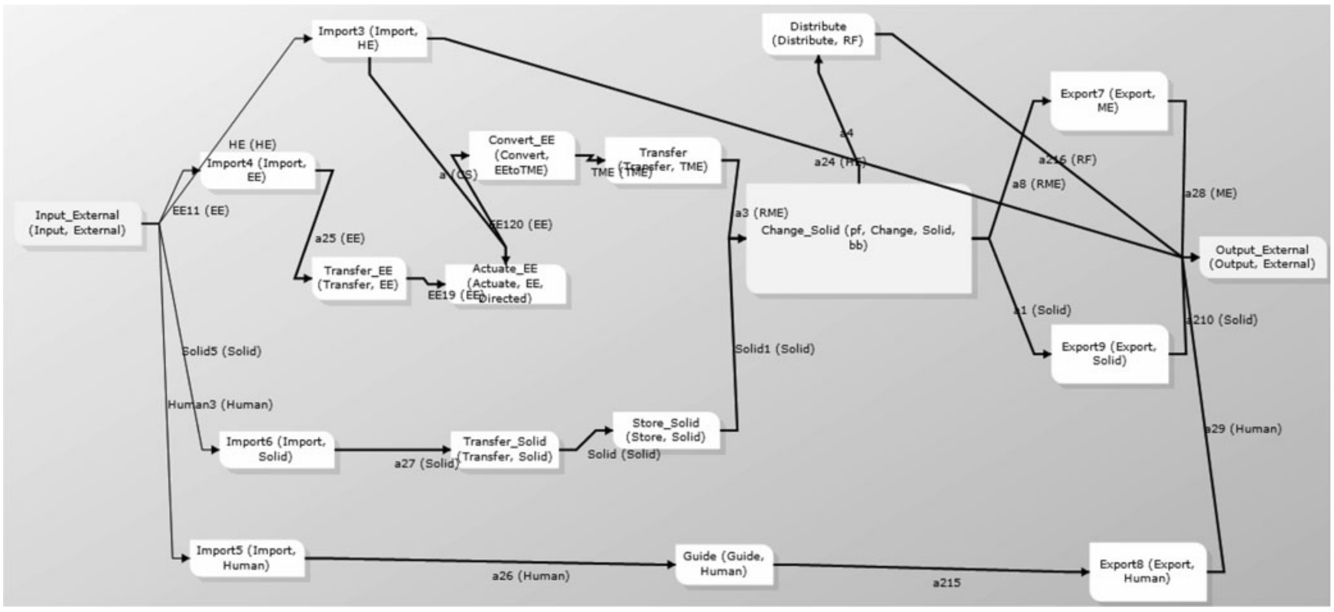

Fig. 8. The first candidate function structure created automatically from the black box shown in Figure 6.

of this part of the research is to present the designer with several different ways to solve the problem. For each possible function the grammar rules are attempting to capture all of the knowledge about this function and which parts can perform this function.

The next step in the automated process is to invoke the automated component selection discussed above as research effort III (see Section 3.3). Current efforts are in place to accomplish this, but a detailed evaluation of component choices is not completed at this time. Fortunately, we are able to approximate the size of the search tree. The average number of components to instantiate is 13 (note that Fig. 10 has 13 components, whereas Fig. 11 has 14). This determines the depth of the tree. The breadth of the tree is 6 (the average number of instantiated components). As a result, the number of possible instantiations is 6^{13} or 13 billion. Taken with the previous search trees, the total number of embodied configurations that result from the single black box is estimated at 20 trillion [1536 CFGs $\times$ (613 instantiations/CFG) $\approx 20 \times 10^{12}$]. This underscores the need for an approach to eliminate many of these branches to focus in on more beneficial solutions. Research effort III accomplished this in the prior implementation through the uniform cost search and specific evaluation functions and functions to penalize systems with incompatible combinations of components. However, to implement this with the current coffee grinder design problem would require us to define functions to evaluate the performance of coffee grinders (cost, noise, uniformity of grain size, etc.) as well as new compatibility functions for the set of the original equipment manufacturer components relevant to

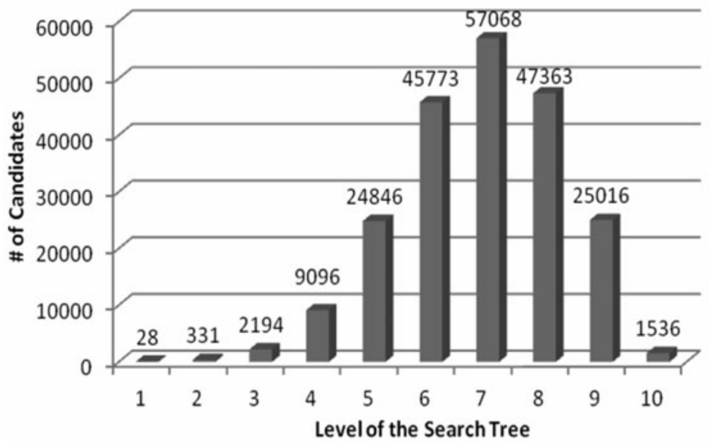

Fig. 9. The bar graph showing the number of candidates at each level.

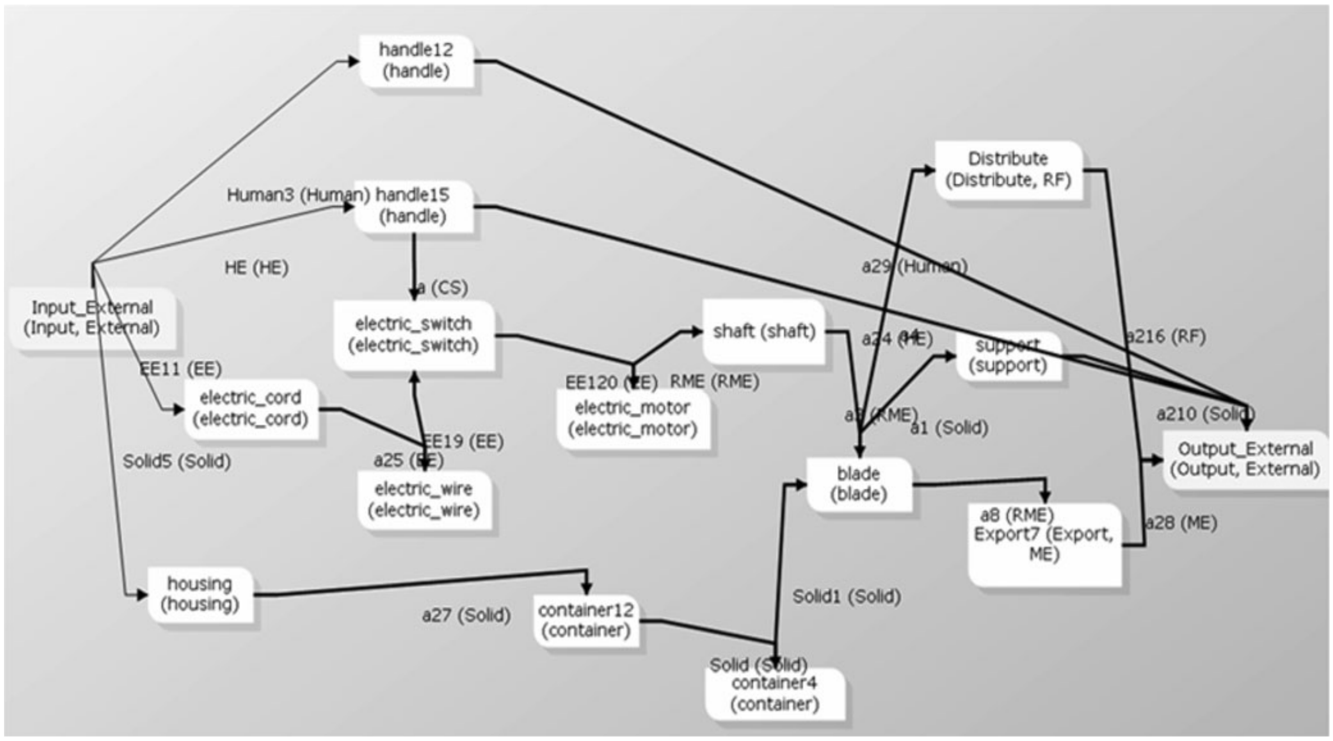
handle12 (handle)
Human3 (Human) handle15 (handle)
HE (HE)
a (CS)
Input_External (Input, External)
electric_switch (electric_switch)
shaft (shaft)
Distribute (Distribute, RF)
a29 (Human)
a24 (HE)
support (support)
a216 (RF)
EE11 (EE)
electric_cord (electric_cord)
EE120 (EE) RME (RME)
electric_motor (electric_motor)
a8 (RME)
a1 (Solid)
a210 (Solid)
Output_External (Output, External)
Solid5 (Solid)
a25 (EE)
EE19 (EE)
electric_wire (electric_wire)
blade (blade)
a8 (RME) Export7 (Export, ME)
a28 (ME)
housing (housing)
a27 (Solid)
container12 (container)
Solid1 (Solid)
Solid (Solid) container4 (container)

Fig. 10. The first candidate CFG created automatically from the function structure shown in Figure 8.

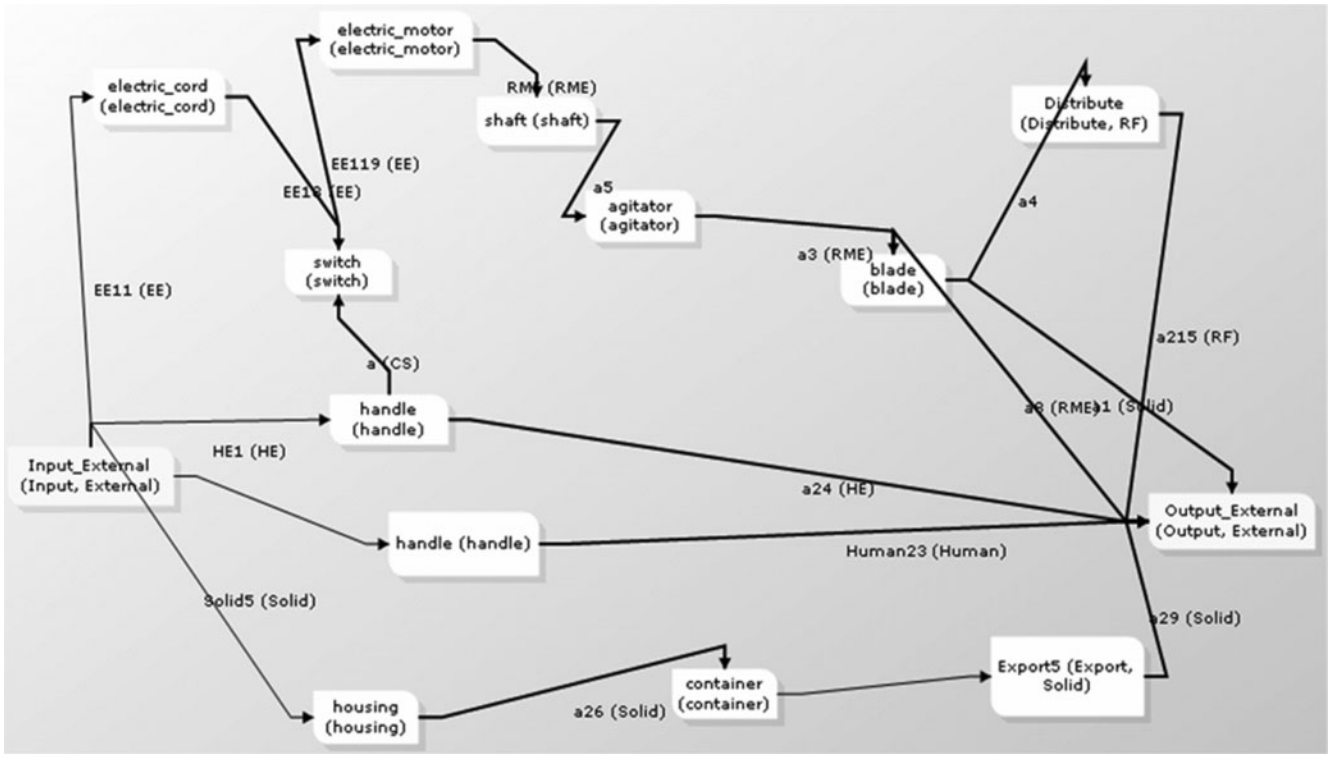

Fig. 11. The second candidate CFG created automatically from the function structure shown in Figure 9.

coffee grinders. This may not be possible without significant and groundbreaking development of simulation environments for predicting noise and grain size uniformity.

6. DISCUSSION

In this paper, three previous research projects are combined to automate the conceptual design process from a black box to a configuration of specific components. This research exercise was accomplished to explore the possibility of using the computer to solve a conceptual design problem through the manipulation of standard graph representations. It is possible that this work could lead to an automated design tool that would present the user with possible connections of components to solve a particular design problem. Because this work lacked an evaluation mechanism, it currently outputs entirely too many solutions for the user to consider. However, combined with a meaningful evaluation, this large number could be reduced to the two or three solutions most useful to the user.

Without the evaluation, 20 trillion possible candidate solutions were found. This large space is not captured explicitly, but rather implicitly in the grammar rules (170 rules total) and the database of real components (300 in all). The grammar affords a representation of the design space as a tree of solutions built from an initial specification. Each transition in the tree corresponds to an application of a rule, thus incrementally building a final design that is represented as one of the leaves of the tree. This process is illustrated in Figure 12. As is evident from the tree, the result of rule applications generates a

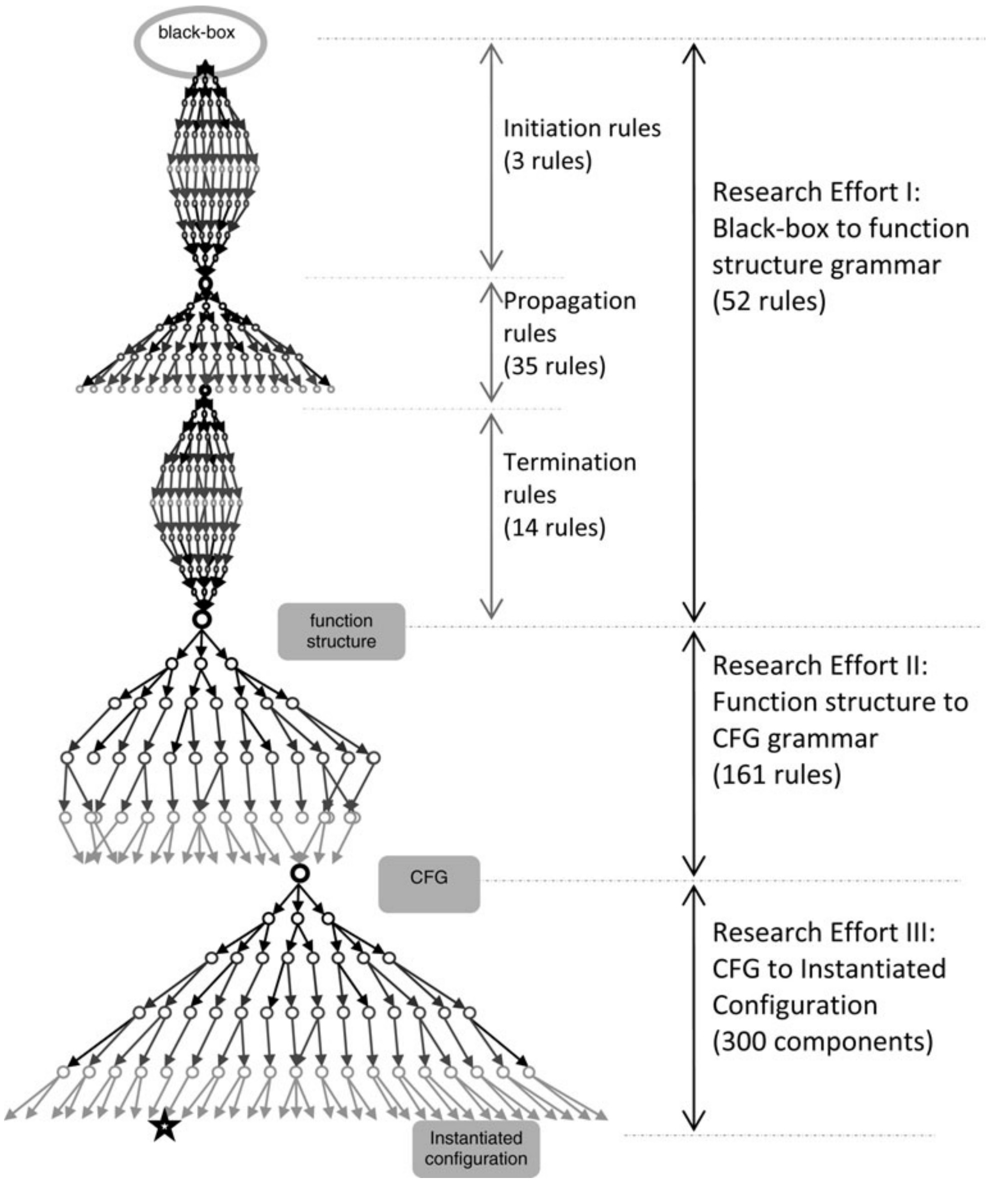
black-box
Initiation rules (3 rules)
Propagation rules (35 rules)
Termination rules (14 rules)
Research Effort I: Black-box to function structure grammar (52 rules)
function structure
Research Effort II: Function structure to CFG grammar (161 rules)
CFG
Research Effort III: CFG to Instantiated Configuration (300 components)
Instantiated configuration

Fig. 12. An illustration of the cascading search trees.

design space that requires navigation techniques to enable a search for a desired or optimal solution. The issue of implementation of the grammar then becomes one of controlled searches through this space of solutions.

The visualization in Figure 12 shows cascading search trees as created by this knowledge base. At each node in the process, a recognition process first retrieves the valid options. The illustration only shows 2 or 3 options per node, but realistically this varies throughout this design process from 1 to 33. Some have claimed that design is essentially a decision-making process, and this is captured by this illustration. With each decision, each commitment to follow a particular branch of the tree, the design process diverges. One can undo decisions to follow other branches or maintain a small diverse set of candidate concepts scattered about the search tree.

Computationally, no single decision is made in the current research; rather, we have taken advantage of a large computational memory to follow every path in the search tree to simply enumerate all possible candidates. This rote approach (BFS) is complete but unmanageable for the final tree in which components are instantiated. Fortunately, this search tree is the first opportunity for us to numerically evaluate the quality of each decision, because real components are being compared and such components have data available about their cost, weight, performance, and so forth. The lack of evaluation limits the computer's ability to decide which option at each stage of the tree is better. Experienced designers can make judgments about which paths to follow in these early stages based on intuition or previous experiences. To take advantage of this, the designer's feedback can be incorporated into the computational system using an interactive approach where the computational system presents solutions to the designer for evaluation. The designer feedback can then be used to assess the value of generated designs and to prune inferior designs from the solution space (Kurtoglu & Campbell, 2009).

Furthermore, the early search trees are complicated by confluence in the rules. Confluence clearly happens and essentially reduces the search tree making it easy to manage, but it is not clear by examining the rules *a priori* how much confluence exists or how to manage it. The tree search algorithms used in this paper include a check for common configurations at each level of the tree to reduce the memory burden; however, this check is time consuming and accounts for 80% of the nearly 5-day span of time required to reach the 1536 candidates at the bottom of the fourth search tree.

The results of this study provide some interesting insight when compared to the human activities in accomplishing the design process. First, the stages of the design process can help to reduce the search space by committing to a best candidate at each level and using that as the seed for the next. These key decision points provide a moment of evaluation and limit the number of solutions needed to be searched in the future. Second, human designers are capable of comparing only a small number of concepts. The current implementation contains many heuristics as stored in the 170 grammar rules, but it is likely that humans collectively know many more, and each likely contains many caveats, exceptions, and useful minutiae. Yet, if all the heuristics about this electromechanical design domain were captured, the number of alternatives at each stage would be even larger. What is required is more information to be stored within the rules: information that defines detailed conditions for when the rule can be applied. Capturing these rules is time-consuming and automated approaches may be possible, but inevitably an experience human designer must define the conditions under which a rule is valid. This is a challenge to the scalability of the approach, and it would be crucial to extend the system to a wider community of users to accomplish a large and useful set of rules for more ambitious design problems.

Third, the design process is vague. Approaches to systemize it like design tools such as creating a black box or a function structure clarify the design process and make it more scientific. Our work has attempted to transition these design tools into an even more rigorous language. The results are promising as the computer is capable of creating a configuration of real components, a coffee grinder in this paper, into a real set of connected components. With more rules and a thorough evaluation of concepts, it now seems possible that the conceptual design process can be solved wholly computationally.

ACKNOWLEDGMENTS

This material is based on work supported by the National Science Foundation under Grants CMMI-0448806 and IIS-0307665. Any opinions, findings, and conclusions or recommendations presented in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

Agarwal, M., & Cagan, J. (1998). A blend of different tastes: the language of coffee makers. *Environment and Planning B: Planning and Design 25(2)*, 205–226.

Bhatta, S., Goel, A., & Prabhakar, S. (1994). Innovation in analogical design: a model-based approach. *Proc. AI in Design*. Dordrecht: Kluwer Academic, pp. 57–74.

Bracewell, R.H., & Sharpe, J.E.E. (1996). Functional description used in computer support for qualitative scheme generation—Schemebuilder. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 10(4)*, 333–345.

Brown, K.N., & Cagan, J. (1997). Optimized process planning by generative simulated annealing. *Artificial Intelligence in Engineering Design, Analysis and Manufacturing 11*, 219–235.

Bryant, C., Stone, R., McAdams, D., Kurtoglu, T., & Campbell, M. (2005). A computational technique for concept generation. *ASME 2005 Int. Design Engineering Technical Conf.*, Long Beach, CA.

Campbell, M., Cagan, J., & Kotovsky, K. (2000). Agent-based synthesis of electro-mechanical design configurations. *Journal of Mechanical Design 122(1)*, 61–69.

Campbell, M.I. (2007). The official GraphSynth Site, University of Texas at Austin. Accessed at http://www.graphsynth.com

Carlson, S.E. (1996). Genetic algorithm attributes for component selection. *Research in Engineering Design 8(1)*, 33–51.

Carlson-Skalak, S., White, M.D., & Teng, Y. (1998). Using an evolutionary algorithm for catalog design. *Research in Engineering Design 10(2)*, 63–83.

Chakrabarti, A., & 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.

Dallaali, M.A., & Premaratne, M. (2004). Double-constrained optimization of optical component selection problem using genetic elitism and double string coding. *Int. Conf. Numerical Simulation of Ootoelectronic Devices '04*.

Hirtz, J., Stone, R., McAdams, D., Szykman, S., & Wood, K. (2002). A functional basis for engineering design: reconciling and evolving previous efforts. *Research in Engineering Design 13(2)*, 65–82.

Hundal, M. (1990). A systematic method for developing function structures, solutions and concept variants. *Mechanism and Machine Theory 25(3)*, 243–256.

Kirschman, C., & Fadel, G. (1998). Classifying functions for mechanical design. *Journal of Mechanical Design, Transactions of the ASME 120(3)*, 475–482.

Kitamura, Y., & Mizoguchi, R. (1999). Metafunctions of artifacts. *Proc. 13th Int. Workshop on Qualitative Reasoning (QR-99)*, Loch Awe, Scotland, pp. 136–145.

Kota, S., & Chiou, S.-J. (1992). Conceptual design of mechanisms based on computational synthesis and simulation of kinematic building blocks. *Research in Engineering Design*, *4*, 75–87.

Kurtoglu, T. (2007). *An computational approach to innovative conceptual design*. PhD dissertation. Austin, TX: University of Texas at Austin Press.

Kurtoglu, T., & Campbell, M. (2009). An evaluation scheme for assessing the worth of automatically generated design alternatives. *Journal of Research in Engineering Design 20*, 59–76.

Kurtoglu, T., Campbell, M., Gonzales, J., Bryant, C., Stone, R., & McAdams, D. (2005). Capturing empirically derived design knowledge for creating conceptual design configurations. *ASME 2005 Int. Design Engineering Technical Conf.*, Long Beach, CA.

Mittal, S., Dym, C., & Morjara, M. (1985). "PRIDE: an expert system for the design of paper handling systems. *IEEE Computer 19(7)*, 102–114.

Navinchandra, D., Sycara, K.P., & Narasimhan, S. (1991). A transformational approach to case-based synthesis. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 5(1)*, 31–45.

Otto, K., & Wood, K. (1997). Conceptual and configuration design of products and assemblies. *ASM Handbook, Materials Selection and Design* (Vol. 20). Materials Park, OH: ASM International.

Otto, K., & Wood, K. (2001). *Product Design: Techniques in Reverse Engineering and New Product Development*. Englewood Cliffs, NJ: Prentice–Hall.

Pahl, G., & Beitz, W. (1996). *Engineering Design: A Systematic Approach*. Berlin: Springer–Verlag.

Qian, L., & Gero, J.S. (1996). Function–behavior–structure paths and their role in analogy-based design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 10*, 289–312.

Schmidt, L., & Cagan, J. (1995). Recursive annealing: a computational model for machine design. *Research in Engineering Design 7(2)*, 102–125.

Shea, K., Cagan, J., & Fenves, S.J. (1997). A shape annealing approach to optimal truss design with dynamic grouping of members. *ASME Journal of Mechanical Design 119(3)*, 388–394.

Sridharan, P., & Campbell, M.I. (2004). A grammar for function structures. *Proc. ASME DETC*, Salt Lake, UT.

Starling, A.C., & Shea, K. (2003). A grammatical approach to computational generation of mechanical clock designs. *Proc. ICED '03 Int. Conf. Engineering Design*, Stockholm, Sweden.

Starling, A.C., & Shea, K. (2005). Virtual synthesizers for mechanical gear systems. *Proc. ICED'05 Int. Conf. Engineering Design*, Melbourne, Australia.

Stone, R., & Wood, K. (1999). Development of a functional basis for design. *Proc. Design Engineering Technical Conf.*, Paper No. DETC99/DTM-8765, Las Vegas, NV.

Subramanian, D., & Wang, C.-S. (1995). Kinematic synthesis with configuration spaces. *Research in Engineering Design 7(3)*, 193–213.

Suh, N. (1990). *The Principles of Design*. New York: Oxford University Press.

Szykman, S., Racz, J., & Sriram, R. (1999). The representation of function in computer-based design. *Proc. Design Engineering Technical Conf., DETC99/DTM-8742*, Las Vegas, NV.

Tamhankar, M.S., & Campbell, M.I. (2007). An intelligent and efficient tree search algorithm for computer-aided component selection. *ASME Design Engineering Technical Conf. Computers in Engineering*, Las Vegas, NV.

Ullman, D. (1995). *The Mechanical Design Process*. New York: McGraw–Hill.

Ulrich, K., & Eppinger, S. (1995). *Product Design and Development*. New York: McGraw–Hill.

Umeda, Y., & Tomiyama, T. (1997). Functional reasoning in design. *IEEE Expert March–April*, 42–48.

Wang, K., & Yan, J. (2002). An analytical approach to functional design. *Proc. ASME Design Engineering Technical Conf.*, Montreal, CA.

Ward, A.C., & Seering, W.P. (1989). The performance of a mechanical design compiler. *ASME, Design Engineering 17*, 89–97.

Wielinga, B.J., & Schreiber, G. (1997). *Configuration design problem solving*. Technical Report, University of Amsterdam, Department of Social Science Informatics.

Williams, B.C. (1990). Interaction-based invention: designing novel devices from first principles. *AAAI-90 Proc., 8th National Conf. Artificial Intelligence*, Vol. 1, pp. 349–356.

Tolga Kurtoglu is a Research Scientist with MCT at the Intelligent Systems Division of the NASA Ames Research Center. He received his PhD in mechanical engineering from the University of Texas at Austin in 2007. Dr. Kurtoglu has published over 30 articles and papers in various journals and conferences and is an active member of ASME, AIAA, AAAI, ASEE, and the Design Society. His research focuses on the development of prognostic and health management technologies for complex systems, model-based diagnosis, computational design tools and optimization, automated reasoning, conceptual design theory, artificial intelligence in design, and risk and reliability-based design.

Albert Swantner is a Design Engineer at Flextronics Medical–Avail in Irving, Texas. He graduated from the University of Texas at Austin with a BS in mechanical engineering in 2007 and an MS in mechanical engineering in May of 2009. His thesis focused on automating and optimizing gear train topologies.

Matthew Campbell is an Associate Professor in the Mechanical Engineering Department at the University of Texas at Austin. He received his PhD in mechanical engineering from Carnegie Mellon University in 2000. Dr. Campbell has been acknowledged with best paper awards at conferences sponsored by the ASME, the ASEE, and the Design Society. His research focuses on computational methods that aid the engineering designer earlier in the design process than traditional optimization would. To date, he has been awarded $1.57 million in research funding, including the CAREER award for research into a generic graph topology optimization method.