Amaresh Chakrabarti
Innovation, Design Study and Sustainability Laboratory (IdeasLab), Centre for Product Design and Manufacturing, Indian Institute of Science, Bangalore, Karnataka, India

Kristina Shea
Virtual Product Development Group, Mechanical Engineering, Technische Universität of München, Garching, Germany

Robert Stone
School of Mechanical, Industrial and Manufacturing Engineering, Oregon State University, Corvallis, Oregon

Jonathan Cagan
Department of Mechanical Engineering, Carnegie-Mellon University, Pittsburgh, PA

Matthew Campbell
Department of Mechanical Engineering, University of Texas at Austin, Austin, TX

Noe Vargas Hernandez
Department of Mechanical Engineering, College of Engineering, The University of Texas at El Paso, El Paso, TX

Kristin L. Wood
Department of Mechanical Engineering, The University of Texas at Austin, Austin, TX

Computer-Based Design Synthesis Research: An Overview

One of the hallmarks of engineering design is the design synthesis phase where the creativity of the designer most prominently comes into play as solutions are generated to meet underlying needs. Over the past decades, methodologies for generating concepts and design solutions have matured to the point that computation-based synthesis provides a means to explore a wider variety of solutions and take over more tedious design tasks. This paper reviews advances in function-based, grammar-based, and analogy-based synthesis approaches and their contributions to computational design synthesis research in the last decade. [DOI: 10.1115/1.3593409]

Keywords: synthesis, search, optimization, grammars, analogy, biomimetics

1 Introduction

Engineering design is the process of satisfying requirements by developing and synthesizing building blocks into meaningful designs that meet the requirements to fulfill needs and desires. Requirements satisfaction depends both on how well requirements are identified, and how well these are applied during the design process [1]. The process and resulting design are at times novel, creative and innovative, and at times routine. Design synthesis is the area of research that focuses on developing guidelines, methods and tools for supporting creation of such solutions. Computer-based design synthesis is important for two reasons: it is sometimes hard to develop novel solutions due to limitations of knowledge or fixation. Here, computers can help designers explore new directions by providing a wider variety of possibilities thereby expanding the range of solutions that are normally considered and, possibly, improving novelty. The other difficulty is the tedium in some design synthesis tasks, e.g., during routine design. In this case, computers can automate tasks, leaving more time for creative activities, and help reduce errors, thus improving value.

Contributed by the CAD/Solid Modeling Committee of ASME for publication in the JOURNAL OF COMPUTING AND INFORMATION SCIENCE IN ENGINEERING. Manuscript received October 5, 2010; final manuscript received February 2, 2011; published online June 15, 2011. Assoc. Editor: J. Shah.

In this paper, three major synthesis themes are reviewed: function-based synthesis, grammar-based synthesis, and analogy-based design. Function-based synthesis focuses on developing representations of a design problem in terms of its functions and producing solutions from functional models. Grammar-based synthesis focuses on developing formal grammars, which contain a design vocabulary and rules that are applied interactively or automatically by computers for transforming initial designs into a wide variety of new designs. Analogy-based design involves identifying analogical knowledge for solving a given design problem and transferring this knowledge to develop solutions, with special focus on case-based design and biologically-inspired design.

2 Function-Based Synthesis

The front end of the conceptual design process has seen few attempts at automation, perhaps due in part to the evolving strategies and methodologies that exist for this phase of design. However, over the past decade, several methodologies have coalesced around the functional decomposition and partial solution manipulation techniques originally introduced by Pahl and Beitz [2], e.g., [3–12]. These methodologies take designers through steps that help decompose a design problem and build conceptual solutions based on the intended, product functionality. Functional modeling

methods abstract the intended functionality of a solution from customer needs, ideally removing designer biases that may be introduced by focusing on specific solutions too early in the design process. This abstraction helps designers generate more complete conceptual solutions and balance design choices among alternative components with the same functionality [2].

Research into the benefits of structured design methods (e.g., [13]) coupled with research into designers' reluctance to use them (e.g., [14,15]) seem to point toward the need for the seemingly tedious stages of systematic design to employ some level of automation to help integrate the benefits of a structured method with the more natural activities of a designer–a need that is most evident during the early phases of conceptual development.

Computational tools for conceptual design do exist, yet these tools often address areas that support aspects such as initial requirements gathering (e.g., organizational tools such as the TikiWiki project [16]), the creation of function structures (e.g., the function grammar tool developed in Ref. [17]), or optimization of well-established concepts (e.g., [18]) rather than the translation of functional requirements into creative solutions.

2.1 Fundamental Developments Supporting Functional Synthesis

2.1.1 Product Function Representations. Function is variously described, the two convergent alternative meanings being device-centric and environment-centric [19,20]. Functional modeling is often considered a fundamental abstraction and a key step in the conceptual design process [5,21–26]. Its application allows design problems to be quickly abstracted without the need to consider potential components, known solution principles or design impossibilities. Functional modeling from a constrained list of computer parseable terms can trace its roots back to value analysis with the work of Miles [23] and Rodenacker [27]. This early work is expanded through the proposal of additional functions by Roth [28], and further formalized through the Koller's proposal of twelve basic functions [29]. At a high level of abstraction, Pahl and Beitz develop a list of five generally valid functions and three flow types [30]. Hundal then proposes a set of six function classes [31], but excludes the flow of information, which is re-added by Little et al., with the functional basis set [32]. Standardized sets of function and flow terms are proposed separately by Szykman et al. [33] and Stone and Wood [34]. These terms are reconciled by Hirtz et al. into the functional basis [35] to form a standard lexicon consisting of two sets of morphemes—one for functions and another for flows.

Beyond functional basis modeling, various parallel functional modeling and associated synthesis techniques have been proposed and explored to aid design. Alternative to the view of developing function models using a vocabulary of general functions independent of artifacts is the complementary view, originated by Hubka and Ernst Eder [7], where functions at a higher level of abstraction can be decomposed further only with the assumption of the means with which to fulfill these functions. It was further demonstrated by Chakrabarti et al., theoretically [36] and empirically [37] that functions cannot be further decomposed, while ensuring both convergence to solutions and solution-neutrality. Function-means tree is the generic name given to this coupled nature of functions and structures [38]; similar concept is used in synthesis of computer programs [39]. A variety of systems have been developed to computationally support synthesis as development of function-means trees, interactively [e.g. 40,41] or automatically [36,42,43].

Either of these approaches requires mapping descriptions of function to descriptions of means to fulfill the function, sometime requiring multiple levels of abstraction to connect the overall function and the final structure. Various models to support functional synthesis have been created, starting from the initial function-structure models [e.g., 39], to transformation-function-organ-structure model [7], transformation-organ-structure model [38], function-behaviour-structure model [44], structure-behavior-function model [45], function-behavior-state model [46], function-environment-behavior-structure model [47], function-behavior-component model using Bond-graphs [48], and the more recent and comprehensive State-Action-Part-Phenomena-Input-organ-Effect (SAPPhIRE) model of causality [49], each with associated synthesis support. More complete reviews can be found in Refs. [36, 50–56].

Design Knowledge Collection and Storage. Designers often develop conceptual designs that draw inspiration from previous design knowledge [57–61]. This inspiration-based approach, a form of design by analogy, is discussed further in Sec. 4. Here, we consider fundamental underpinnings that allow functional knowledge about a product or artifact to be collected and stored to support knowledge reuse.

The main objective of using a design repository is to facilitate storage and retrieval of design knowledge at various levels of abstraction, from form to architecture to function, during the product development process. Building on the functional basis representation, a prototypical design repository–following the National Institute of Standards and Technology (NIST)-proposed format–has been developed [62–64] to support design archival and reuse–essentially within-domain design by analogy. Currently, the repository is housed within the Design Engineering Lab at Oregon State University, and contains design information for approximately 150 consumer-based electro-mechanical products. The repository currently follows the NIST schema, and identifies artifact-, function-, failure-, physical-, performance-, sensory-, and media-related information for each product in XML format [65]. The variety in levels of abstraction and types of design information provide innovative ways to approach design. Initially, artifact information in the repository was recorded in spreadsheets in the form of a collection of files of bills of materials, function component matrices, and design structure matrices. This information was migrated to a more rigorously defined database.

More recently, Oregon State has partnered with UT-Austin [66,67], Penn State [65], Virginia Tech, Bucknell [68], University of Buffalo, Drexel [69], and Texas A&M to expand the types of design information and breadth of design tool features within the repository. The design repository serves as a hub for designers for information exchange and design generation tools. Information entry and retrieval occurs within a standalone application [70] (see http://designengineeringlab.org/repositoryEntry/), while information retrieval occurs over the Internet through the design repository web portal (see http://repository.designengineeringlab.org/). The infrastructure supporting these two applications is the design repository information ontology [65]. The ontology describes what types of design information can be stored, relationships among those elements, and the extensibility of including new and additional types of design information.

2.2 Computer-Aided Functional Synthesis. Computerized concept generation techniques, spanning the broad automatic input topics of knowledge representation and reasoning, promise faster realization of potential design solutions based upon previously known products and implementations. While the area of automated concept generation has made great strides in recent years, most methods still require the user to indicate desired functionality. Using functional descriptions has been shown to help engineers stray away from pre-trained ideas of how a product or device would look and operate [4]. Two of the automated function-based synthesis methods under development today rely solely on the user's ability to develop functional descriptions of their desired product. Both these methods make use of a repository of design information including component connection and component functionality. A component naming taxonomy to classify product artifacts was formulated to standardize the output of the automated function-based synthesis methods [71]. Each artifact is classified under a specific component name according to its primary functionality.

The recent foundations for concept generation through computational reasoning have been based on formalisms for describing

function or purpose in engineering design. Some results from this research include automated morphological matrix generation from the design repository [72,73], more expansive overall concept generation algorithms based on the empirical knowledge of function-component connections in the form of relational matrices [64,66] and graph grammar rules [74,75] (which are detailed further in Sec. 3) that, when combined with a search mechanism, automatically creates conceptual designs. These automated concept generation algorithms give designers the ability to quickly generate concepts based on knowledge stored in the design repository. The two complementary methods both rely on repository information, utilize matrix manipulation and graph-grammar rules. The matrix manipulation-based concept generation method, morphological evaluation machine and interactive conceptualizer (MEMIC), translates the input function structure into a matrix form that describes the adjacency between functions. This input undergoes a series of matrix multiplications that map functionality to solutions (components) and filters out infeasible component-to-component connections based on the repository data. The output of MEMIC is a set of concept variants that solve the input functionality [14].

From a perspective different from the function modeling approach discussed above, a number of research efforts have sought to establish a generic computational scheme for electromechanical design, including those for sensor synthesis [76]. While these methods do not all utilize as formal a functional language as expert human designers tend to use, such approaches have been shown to successfully synthesize new electromechanical configurations. These methods use a variety of computer techniques including case-based reasoning [77], constraint programming [78], qualitative symbolic algebra [79], or geometric algebras [45,80]. One of the most historically significant among these is PRIDE [81], which uses expert systems techniques for design of paper handling systems.

In the approaches reviewed, the repeating refrain is that computational synthesis approaches tend to be computationally complex, often producing an overwhelming number of concept variants that are impossible to explore without support; efficient algorihms and appropriate methods are needed to realistically identify and explore all feasible solutions. Efficient algorithms to generate solutions [82], Heuristic techniques [83], and side effects detection techniques [84] to prune solutions, novel visualisation techniques to explore representative cases in large solution spaces [85], and structure sharing to create resource-effective solutions [86,87] are some of the potential approaches for dealing with creation and handling of large, realistic solution spaces. As shown by Fricke [88], balanced search, e.g. the ideal approach proposed in Ref. [89] where synthesis progresses through multiple divergence and convergence steps, is a possible answer to this. Furthermore, those results show that subtle challenges in a given design problem may not always be captured in the specification of initial function, and thus many results were not relevant to the user's needs [90,91]. Consequently, the proof of concept designer preference modeler [92] was created to find within the large set of results which concepts were most meaningful to the designer. By ranking select concepts, the search mechanism learns what aspects of the concept the user prefers, and seeks solutions that maximize the preferred aspects.

3 Design Synthesis Using Generative Grammars

An important aspect of conceptual design is the generation of a wide range of alternative designs that encourage designers to "think outside the box". Generative grammars are used to computationally encode knowledge about creating designs, either a certain class or style, which can be used to rapidly generate design alternatives. Both standard and novel solutions can be generated, that often go beyond a designer's normal approach to expand their thinking and spark creativity. Generative grammars can also be used to better understand solution spaces, including the complex constraints that often define them, also as they evolve and change over time under influences from other domains, e.g., changing customer desires and manufacturing capabilities.

Engineering grammars are a class of production systems that capture design knowledge by defining a vocabulary and rule-set, which operates over the vocabulary, to generate a set of designs, called the design language. While many grammar types exist [93], the most prevalent in engineering design are graph and spatial grammars. The term spatial grammars is used to describe all kinds of grammars that define languages of shape, e.g. string grammars, set grammars, shape grammars, and graph grammars [94].

An engineering design grammar is developed and applied using the following main steps:

1. Determine representation, e.g. string, set, shape or graph.
2. Define vocabulary.
3. Define grammar rules.
4. Define initial design.
5. Generate designs within the language, which includes recognizing where and how a rule applies and applying it to generate a new design.
6. Modify the language, e.g., vocabulary and rules, and return to step 4.

The typical view is to define a left-hand side (LHS) of rule conditions that define when the rule is valid and a right-hand side (RHS) of rule modifications, which can involve adding, subtracting, or modifying objects. The LHS is matched to a sub-graph, or sub-shape, in the working graph, or shape, which is replaced by the defined graph, or shape, in the RHS of the rule. This "if-then", or LHS→RHS format, is common to all grammar formalisms.

This section provides a review of both spatial and graph grammars focusing on major advances in the last 10 years. Many engineering design grammars have been developed that capture the language of different domains; see Ref. [95] for a review. However, a main roadblock to achieving wider impact, especially in conceptual design, has been to support designers in the iterative development of a grammar, without having to program it directly, and its application to rapidly generate alternative designs. Thus, two focal points of the review include recent advances in providing grammar interpreters as well as automatic learning of grammars. A grammar interpreter is defined as a software program that interactively and graphically supports the steps defined above, without the need for extensive programming.

3.1 Graph Grammars.

Graph grammars were originally described as graph rewriting systems, and built on a rich body of work in the 1950's on string grammars [96–98]. Over the past 50 years, the advances in expert-systems, object-oriented programming and even graphical-user interfaces have brought graph grammars to a level ripe for capturing real and complex engineering design rules.

One example is GrGen, which is an application-independent, open-source software framework for the implementation and development of graph grammars. It provides for the development of an expressive, turing-complete rule language with extremely fast rule execution [99]. Integrating GrGen, the open-source software BOOGIE takes an object-oriented approach to defining a meta-model, or vocabulary, [100] to provide a hierarchical structure for vocabulary definition and enhance its extendibility and reusability. The meta-model incorporates different levels of abstraction, i.e., function-behavior-structure, and defines interconnections between vocabulary, both within one level and between levels, based on the definition of ports. The definition of ports enables the use of generic rules at all levels, which are independent from the vocabulary definition, through port matching, in addition to allowing the graphical description of application specific graph rules. A recent application has been to the synthesis of hybrid power-train architectures from a high-level function model down to a structure model [101].

The Design Compiler 43 [102] is a general, Eclipse-based, platform for solving design synthesis problems based on the graph grammars. The knowledge and procedure to solve design

problems can be formalized in a graph grammar where rules are matched using subgraph recognition based on regular expressions. The key advantage is the many interfaces provided to transform design graphs into analysis models, allowing for integration with common tools, e.g. CAD and simulation. A recent application is to the design of satellites [103].

The graph grammar software GraphSynth, has been developed and successfully used as a basis for automatically synthesizing a number of engineering designs including sheet metal parts [104], conceptual electromechanical designs [74], and disassembly sequences [105]. Through a rich graphical user-interface, one can create graphs, rules, and sets of rules that define a language of graph topologies. GraphSynth includes a facility to incorporate search and parametric optimization routines to automate the creation of designs. In one instance, design repository data are used to formulate "grammar rules", currently numbering over 150, to transform a function structure into a graph of connected components, referred to as a component flow graph (CFG).

A main issue with applying graph grammars in engineering design is that there is no commonly agreed language for writing them, i.e., graph grammars, containing rich design knowledge, and resulting designs are not transferable between systems. Generally, each research group has their own custom representation and implementation. Research focused on formal, standardized languages, e.g., SysML or Moka ML [106], as well as using graph grammars for transforming one model to another would facilitate an exchange of grammars between researchers and users. While this area of research in computer science is mature, research is needed to transfer and extend it to meet the complex knowledge representation requirements in engineering design. One recent approach in this direction is that of [107] where the formal modeling language SysML is used to define a set of components, including structure, dynamic behavior and cost, for which the definition of graph grammar rules is carried out based on the meta-language MOF. This is successfully applied to generate fluid-power circuits, where rule application is based on probabilistic selection of rules.

3.2 Spatial Grammars. Spatial grammars have also been widely used for design synthesis and conceptual design. Shape rules are defined in the form A→B, where A and B are both shapes in the vocabulary. Shapes can be represented by strings, sets, e.g., geometric primitives, shapes defined through maximal line representations and also graphs, e.g. boundary representations. Here, the matching of rules is different to graph grammars in that when detecting sub-shape A in working shape C, Euclidean transformations are used, e.g. translation and rotation, to find more possible matches. Further, parametric spatial grammars allow for more generic shape rules to be described where parameter values are then also determined in the matching process. Recent reviews of shape grammar implementations and interpreters can be found in Refs. [108, 109]. While interpreters for graph grammars are well underway stemming from the strong foundation coming from computer science research, similar level shape grammar interpreters generally lag behind.

An interactive, 3D shape grammar interpreter has recently been developed by Hoisl and Shea [109] that is integrated with an open-source CAD system and geometric modeling kernel. It takes a set grammar approach defining a vocabulary of parameterized primitives with which both parametric and nonparametric rules can be developed graphically, making use of common CAD functions for creating and editing geometric objects. To support automatic rule application, it provides automatic matching of the LHS of rules and automatic application of rules including arbitrary rotations and translations in combination with assigning parameter values and adhering to defined parametric relations. The system has been successfully applied to generate vehicle wheel rims and cooling fin designs. While limited to 3D primitives, it is a step toward providing a general shape grammar interpreter within a familiar CAD environment.

Shape grammars are often known most for their use of a maximal line representation to enable the detection of sub-shapes, that is shapes that emerge through shape calculation but are not explicitly represented, e.g., as when using a set grammar. For conceptual design, recognition and transformation of emergent shapes is an ultimate goal as it enables a wider variety of shape designs to be generated, some creative even, that a designer would or could not produce by hand. A leading system for 3D shape grammars [108] uses a maximal line representation approach that can handle straight and curvilinear basic elements in 3D space. The shapes and rules are created and edited in an external text file and applied to generate wireframelike shapes. The matching of the LHS, including sub-shape recognition, is carried out automatically where the transformation is given by the user.

An engineering approach to parametric subshape recognition was introduced by McCormack and Cagan [110,111], achieved through a decomposition of shapes into a hierarchy of subshapes ordered by their decreasing restrictions. Instances of each subshape are individually located in the design shape and then reconstructed to form an instance of the entire shape. The basis for the hierarchy of subshapes can be specified by the designer or based on default spatial relations that come from architectural and engineering knowledge. The levels of the hierarchy are defined so that the most constrained lines of a shape are those lines that the designer intended exactly. The less constrained segments require more extensive search but the more specific instances have been filtered out already, reducing search requirements. A two-step process that uses Bézier curves enables curve-based matches as well.

Another key recent development in the area of subshape detection is reported in Ref. [112] who take a different approach by recognizing that the problem is similar to that of object recognition in computer vision. Their main contribution is a pixel-based, 2D representation that matches the LHS of a rule to a working shape by determining the visual similarity between the two using the Hausdorff distance, which is a distance metric for point sets. Since the method is not based on an explicit geometric model, all shapes that can be represented through pixels can be considered, coming from CAD models or sketch-based input. While this is the main advantage of the approach, which also allows inexact matches using a tolerance variable, a design can get "messy" after applying several rules making further rule applications not possible after much iteration.

Genesis is currently the only known commercially used spatial grammar system implemented at Boeing [113]. It is based on the original shape grammar system that was developed to generate alternative Queen Anne style houses [114]. It is used to support the interactive generation of tubing designs in aircrafts to enable engineers to explore solution spaces as well as evaluate, compare and merge design alternatives. A 3D boundary solid shape representation is extended to include hierarchical assembly structures and interfaces, part classification and functional schematics. Design rules are formulated through matching conditions and design transformations, rather than as replacement rules.

3.3 Learning Generative Grammars. An emerging area of importance is the learning of grammar rules. In practice today, grammar rules must be "knowledge engineered", namely a person must determine what the important features are for design generation and how to formulate the rules. Engineering grammars would be more readily applied if rules could be learned and adapted on their own.

Yogev et al. [115] evolved rules within the DNA or genome of the genetic structure embedded within an initial cell. The evolution modifies the sequence of basic rules creating meta-rules that dictate how a structure is configured or built. The meta-rule is influenced by environmental conditions and mutations in the rule sequence itself. This work is demonstrated on the design of a structure that evolves to an optimum configuration. The advantage of this approach is that sophisticated rules can evolve from smaller rules based on the environmental conditions.

A different approach to rule learning has been developed for use when the sample or class of products exist from which rules should be learned to generate designs primarily within that class (although the resulting rules can be used outside of the given parametric range as well). Orsborn et al. [116,117] introduced a statistical approach to rule generation whereby a data set of products (in their case vehicles) was analyzed by principal component analysis (PCA) to determine similarities and differences between products in the data set. The analysis is based on a general curve-based representation for the class of products, where the curve control points are the data that are analyzed. Each successive component of the PCA provides the most to least variation in a given dimension of a product. Rules are built based on the curves highlighted in the analysis. Of note, the rules are based on what actually has the most variation, not what people describe as discrete components in a product.

Both of these approaches were successful and show promising directions for rule learning. This is clearly a ripe area for future research, an area that could significantly impact the breadth of application and use of engineering grammars for conceptual design.

3.4 Key Issues for Generative Grammars. At a recent conference workshop (see http://www2.mech-eng.leeds.ac.uk/users/men6am/DCC10-SG-Implementation-Workshop.htm), several key issues were identified to take shape grammars from theory to useable software. These issues apply to all generative grammars and are summarized here: (1) supporting designers to articulate grammars (i.e., vocabulary and rules) in software implementations, (2) defining ways to evaluate implementations, including identifying a set of benchmark problems, (3) better integration of generative grammar implementations with other software, e.g., CAD and CAE and (4) more methodological support for users in the process of defining a grammar since this process can also lead to better understanding of a design problem.

Stemming from the last point, a final issue that needs further attention in the research community relates to education, which has also been considered previously by Knight [118]. Generative grammars are taught only at a relatively small, although growing, number of universities. However, they have great potential since their design and application requires formalization of design knowledge and developing a sound description of the problem to generate potential solutions. Further, they can foster design exploration and better understanding of the solution space. To support education, good software implementations and grammar interpreters are one prerequisite. The challenge is also to foster interest in the area and the building of the right mix of technical, spatial, and intellectual abilities for their effective use.

4 Design by Analogy

Whether through formal or informal methods, designers often develop conceptual designs that draw inspiration from previous design knowledge [57–61]. However, this inspiration approach, a form of analogy-based design, still lacks widespread computational support. Hindering its application is the fact that analogies are difficult to retrieve from memory [119]. This section discusses three overlapping research areas that are centered around analogy: analogy based design (ABD), case based design (CBD), and biologically inspired design (BID). ABD provides the broad umbrella, where designing involves identifying and transferring relevant knowledge, from the same or different domains, to solve design problems. CBD is a sub-area within case based reasoning (CBR), and solves problems by primarily focusing on identifying and adapting knowledge from within domain. BID solves problems by identifying and transferring knowledge from biological domain.

4.1 Analogy Based Design. Wikipedia describes analogy "as a cognitive process of transferring information from a particular subject (the analogue or source) to another particular subject (the target), and a linguistic expression corresponding to such a process." Analogy is valuable for design; one of the first ABD methods was Gordon [120]. Analogical reasoning is traditionally categorized in terms of *why*, *what*, *how* and *when* [121], focusing on the content of knowledge that makes analogical transfer feasible, describing different types of analogies along these dimensions. In ABD, the *why* question focuses on the task for which analogy is used; the *what* question on the content of knowledge transferred; the *how* question on the methods for reminding and transfer; and the *when* question on strategic control of processing.

Analogical transfer requires the use of generic abstractions, which express the structure of relationships between generic types of objects and processes [121]. For a design process to be analogical, the knowledge transferred from a source case to the target problem must be an abstract relationship between objects and not simply an object attribute [122]. As an example of the necessary abstraction, the mechanism of structure-mapping [57,122] is essentially independent of task or domain, size of problem or timing of problem solving, content of knowledge, or modality of knowledge representation. ABD involves learning and transfer of these abstractions from one design situation to another, where the abstractions specify the structure of relations among the elements of a design problem, solution, domain, or strategy, and where transfer can occur to fulfill any design task in the new situation.

To answer the how question of methods of analogical transfer, case-based methods are common for within-domain analogies, see Sec. 4.2. For cross-domain analogical transfer in computational design, various models such as schema-based model [123] is used in which knowledge is transferred from a source case to a target problem by abstracting the solution schema. The abstract problem schema serves as a retrieval cue for finding a solution schema that provides a solution to the target problem. In computational design, the IDEAL system [61,124] (see further below in this section) uses this method for the conceptual design.

Various descriptive studies are undertaken to understand the ABD process. Gero and Maher [125] studied its cognitive processes; Davies and Goel [126] studied visual aspects of analogical transfer; Tseng et al. [127] identified the types of analogies that impact design creativity; McAdams and Wood [128] developed a similarity metric for comparing designs produced by analogy; Wiratunga et al. [129] studied learning strategies used in adapting design cases; Christensen and Schunn [60] identified the functions served by analogy in designing novel design concepts.

A variety of systems for design by analogy are developed for problem formulation and function analysis, mainly using natural language processing e.g. [130–133]. It is also used for improvement of designs for characteristics other than function, e.g., For instance, Balazs and Brown [134] developed a computational approach for design simplification.

The biggest chunk of work in this area focuses on using analogy to fulfill intended functionality. Goel [121] speaks of three distinct theories of analogical design: based on Syn [135], Dssua [136], and Ideal [137]; they, respectively, illustrate three kinds of generic abstractions in analogy-based creative design: design concepts, design prototypes, and design patterns.

Syn is a module within a CAD environment for assisting architects in designing spatial layouts of buildings; the basic process classifies a given design problem into a library of stored design concepts, retrieves the best matching concept, and instantiates the concept in the context of the problem to generate a candidate solution.

Dssua is an interactive system to address mechanical design problems within architectural designs. It stores knowledge of familiar designs as design prototypes [43], using the function-behavior-structure model. Transfer is based on similarity between the structures of the dependency graphs abstracted from the design prototypes and the initial solutions specified as part of design problems. Further, Kulinski and Gero [138] proposed a model for situated analogy in design.

Ideal uses model based analogy for conceptual design of engineering devices. It contains several kinds of domain knowledge: design analogues, design patterns, design concepts, generic design components, and generic domain substances. Design analogues are expressed using structure-behavior-function model. For transfer, IDEAL develops adaptation goals in terms of the differences between the functions of the new and the retrieved designs, and refines the new design by reducing these differences, thereby generating new designs through analogical transfer of design patterns acquired from earlier design episodes.

4.2 Case Based Design. CBR involves "adapting old solutions to meet new demands, using old cases to explain new situations, using old cases to critique new solutions, or reasoning from precedents to interpret a new situation." Kolodner [139]. CBR is a limiting case of analogical reasoning in which the question of what to transfer translates into that of what to modify [121]. CBD is "the process of creating a new design solution by adapting and/or combining previous design solutions" [140].

Many reviews of CBR exist, e.g., [139–144]. CBR techniques are a progression from knowledge based systems, with the goal of alleviating some of their traditional issues with knowledge elicitation, implementation and maintenance [145].

In case-based methods, a designer attempting to solve a target problem is first reminded of a similar source problem for which the solution is known. Then the target problem can be solved by transferring and adapting the source problem solution for the target problem e.g., [5,22,24]. Aamodt and Plaza [142] describe CBR as a cyclical process comprising four REs: RETRIEVE similar cases; REUSE the cases to solve the problem; REVISE the proposed solution if necessary, and RETAIN the new solution as part of a new case. For CBD, the first three steps translate to *propose*, *critique,* and *modify* [146]. This cycle rarely occurs without human intervention: CBR tools primarily support case retrieval and reuse, while case revision or adaptation is mainly carried out by humans [146].

The work on scripts by Schank and Abelson [147] probably started CBR. Schank produced a cognitive model for CBR based on the concept of dynamic memory; Kolodner [148,149] used this to develop the first CBR system CYRUS. Some of the early CBD systems [140] are CYCLOPS [150] and STRUPLE [151].

A case is a contextualized piece of knowledge representing an experience, which typically contains the problem that describes the state of the world when the case occurred, the solution that states the derived solution to that problem, and/or the outcome which describes the state of the world after the case occurred [145]. Case indexing involves assigning indices to cases to facilitate retrieval. The case-base should be organized to provide both semantic richness of cases and their indices, and methods that simplify case access and retrieval. These methods are called case memory models; the two most influential are the dynamic memory model [148,149,152], and the category-exemplar model [153]. Given the description of a problem, retrieval involves using the indices in the case-memory to retrieve cases similar to the current problem. Retrieval algorithms include: heuristic/analogical [154], serial search [155], hierarchical search [156], and simulated parallel search [157]. Adaptation involves modifying the solution stored in the retrieved case to the needs of the current case; Avramenko and Kraslawski [158] summarize two approaches–structural adaptation and derivational adaptation, and a various adaptation techniques ranging from no adaptation to case-based substitution. Kolodner [139] specifies two further methods: substitution methods and special purpose methods. However, automated case adaptation is particularly difficult.

4.3 Biologically Inspired Design. Biological systems resource-effectively fulfill various tasks within a variety of environments and constraints; many of these are similar to those in engineering design. Therefore, biological systems offer a rich source of inspiration for design [159]. BID (also termed biomimicry or biomimetics) is an emerging area in which ABD is carried out with biological systems as analogues. Wikipedia defines biomimetics as the examination of nature, its models, systems, processes, and elements to emulate or take inspiration from in order to solve human problems.

Biologically inspired designs have traditionally been an outcome of individual interest, accidental exposure, or systematic study [156]. Cross-domain inspiration is found to be a valuable source for analogical transfer [160]. However, understanding the BID process and supporting this systematically is only beginning to be researched.

Descriptive BID studies are relatively few. Helms et al. [161] analyzed processes in BID projects, and found that both solutions and problem decompositions were transferred in BID. Compound solutions were generated using both analogy and problem decomposition Vattam et al. [162]. Arguing that current understanding of the analogical basis of BID is limited; Vattam et al. [163] analyzed BID processes to develop a model of using creative analogies in BID. Sartori et al. [164] analyzed biomimetic design cases to identify a generic set of abstraction levels at which biomimetic transfer occur in design.

Prescriptive studies in BID include developing biomimetic processes, databases, guidelines, tools, and applications. Hill [165,166] proposed an orientation model for biomimetics that involved goal setting and solution identification. Schild et al. [167] proposed a systematic approach for identifying biological analogues for a given problem. Gramann [168] proposed a biomimetic process to support technical problem solving. Based on analysis of such models, Sartori et al. [165] proposed a generic model of the biomimetic process with these steps: formulate search objectives; search for biological analogues; analyze them; and transfer.

Databases of biological systems developed to support BID include: catalogue sheets [165,166] that capture knowledge about biological structures and functions; a database of biological effects structured using TRIZ [169,170]; a database of biological systems structured using the SAPPhIRE model of causality [49]; function-based approaches [171], SBF based approaches [172], and approaches using reverse engineering and ontologies [173,174].

A contrasting approach is proposed by Hacco and Shu [e.g., 175] where methods are developed to use natural language processing to extract biological information available in documents in natural-language format; designers can apply analogical reasoning to transfer this knowledge to the target domain, see [176] for a review of these.

According to Chakrabarti and Shu [159], both these approaches have their (de-)merits. While the latter approach requires undertaking neither the structuring effort nor the effort of populating a database using this structure, effort must be invested in developing appropriate search strategies for locating meaningful information from the plethora of existing knowledge sources available in natural-language format. The former approaches, in contrast, require investing substantial effort into prestructuring information, and entering a meaningful quantity of information using that structure, for subsequent benefits of easier and more focused search.

Guidelines are proposed: for biomimetic transfer [165] and composition of biological systems [161]. Several software tools are developed, e.g. IDEA-INSPIRE [49,177,178], DANE [172], etc. Various biomimetic applications are developed, e.g., an Eel-like swimming robot [179] and a millipede-inspired multi-tracked rover with high obstacle climbing performance [180].

4.4 Key Challenges in Design by Analogy. Two major challenges remain in computational design by analogy research: how to (automatically) identify what analogical material is relevant for a given design problem, and how to (automatically) transfer these to solve the problem. In the context of biologically inspired design, developing a meta-language with which the source and target domains can be expressed and mapped remains a major challenge.

5 Discussion and Conclusions

A major goal of design synthesis is to support creation of a large number of alternatives of high value. The review shows a wide variety of methods proposed to help with generation of alternative problem specifications, a wider variety of design alternatives to fulfill these, and some methods for their evaluation and improvement.

Several major sub-themes emerge within each theme. Function-based synthesis methods cluster around the notion of common, generic functions for problem decomposition, and those that bootstrap function and means to move from abstract, overall functions to concrete, overall solutions. Grammar based synthesis methods have two major sub-themes: graph grammars and spatial grammars, both of which define languages of design that can be used to interactively or automatically generate routine and novel design solutions. They intersect with function-based synthesis methods since graph grammars in particular are an effective means to generate concepts starting from a high-level function model down to product structure and component generation. At the component-level, spatial grammars can be integrated to generate both standard and complex forms within a specified design language. Analogy-based design focuses on case-based reasoning (mainly in-domain) and analogical reasoning (mainly cross-domain) approaches, where the latter draws inspiration increasing in the emergent theme of biologically inspired design. It is analogy-based design where the function- and grammar-based design approaches may hold the most promise at improving the knowledge transfer from source to target case.

In the last decades, much theory has been laid and research methods developed for computer-based design synthesis, as reviewed in this paper. However, major challenges remain, which are summarized here to drive further research:

- A commonly accepted terminology and language for the field is still needed to foster exchange of research models, methods and results.
- In biologically inspired design specifically, developing a meta-language for expressing both source and target domains remains a major challenge, as does the structuring analogical materials.
- While analogy-based design has taken great strides in retrieval research, adaptation still remains a major challenge, the difficulty accentuated by the context-specificity of the possible adaptation options.
- Exploration of large solution spaces created by design synthesis methods can be overwhelming, and further research is needed on appropriate and effective search, evaluation and optimization methods.
- Supporting designers to formalize their knowledge of synthesis in software implementations is a long standing issue shared with early expert systems and current knowledge management.
- Rigorous means for evaluating synthesis systems, including common benchmark cases and criteria for assessing quality, is needed.
- Synthesis systems need to be scaled-up to support problem solving at the levels of complexity expected in practice.
- Finally, a better understanding is needed for how synthesis systems can be integrated in current design processes in practice and with current toolsets.

Research is currently underway by the authors and the research community at large to address these in order to push computational design synthesis toward use in everyday design practice.

Acknowledgment

The second author thanks the Deutsche Forschungsgesellschaft (DFG) for funding their research in this area as a part of the collaborative research centre "Managing cycles in innovation processes – Integrated development of product service systems based on technical products" (SFB 768).

References

[1] Chakrabarti, A., Morgenstern, S., and Knaab, H., 2004, "Identification and Application of Requirements and Their Application on the Design Process: A Protocol Study," Res. Eng. Des., **15**(1), pp. 22–39.
[2] Pahl, G., and Beitz, W., *Engineering Design: A Systematic Approach* (Springer Verlag, London, 1996).
[3] Otto, K., and Wood, K. L., *Product Design: Techniques in Reverse Engineering, Systematic Design, and New Product Development* (Prentice-Hall, New York, 2001).
[4] Ullman, D. G., 2002, *The Mechanical Design Process*, 3rd ed., McGraw-Hill, New York.
[5] Ulrich, K. T. and Eppinger, S. D., 2004, *Product Design and Development*, 3rd ed., McGraw-Hill/Irwin, Boston.
[6] Cutherell, D., 1996, "Chapter 16: Product Architecture," *The PDMA Handbook of New Product Development*, M. Rosenau, Jr., ed., Wiley, New York.
[7] Hubka, V., and Ernst Eder, W., *Theory of Technical Systems* (Springer-Verlag, Berlin, 1984).
[8] Otto, K., and Wood, K., 1996, "A Reverse Engineering and Redesign Methodology for Product Evolution," Proceedings of the 1996 ASME Design Theory and Methodology Conference, 96-DETC/DTM-1523, Irvine, CA.
[9] Otto, K., and Wood, K., 1997, "Conceptual and Configuration Design of Products and Assemblies," *ASM Handbook, Materials Selection and Design*, *ASM International*.
[10] Pimmler, T., and Eppinger, S., 1994, "Integration Analysis of Product Decompositions," Proceedings of the ASME Design Theory and Methodology Conference, DE-Vol. 68.
[11] Schmidt, L., and Cagan, J., 1995, "Recursive Annealing: A Computational Model for Machine Design," Res. Eng. Des., **7**(2), pp. 102–125.
[12] Shimomura, Y., Tanigawa, S., Takeda, H., Umeda, Y., and Tomiyama, T., 1996, "Functional Evaluation Based on Function Content," Proceedings of the 1996 ASME Design Theory and Methodology Conference, 96-DETC/DTM-1532, Irvine, CA.
[13] Radcliffe, D., and Lee, T. Y., 1989, "Design Methods Used by Undergraduate Engineering Students," Des. Stud., **10**(4), pp. 199–207.
[14] Cross, N., 1994, *Engineering Design Methods: Strategies for Product Design*, 2nd ed., John Wiley and Sons, Chichester, UK.
[15] Ivashkov, M., 2004, "ACCEL: A Tool Supporting Concept Generation in the Early Design Phase," Ph. D. thesis, The Eindhoven University of Technology, Eindhoven, The Netherlands.
[16] Wodehouse, A., Grierson, H., Ion, W. J., Juster, N., Lynn, A., and Stone, A. L., 2004, "Tikiwiki: A Tool to Support Engineering Design Students in Concept Generation," International Engineering and Product Design Education Conference, Delft, Netherlands.
[17] Sridharan, P., and Campbell, M., 2005, "A Study on the Grammatical Construction of Function Structures," Artif. Intell. Eng. Des. Anal. Manuf., **19**(3), pp. 139–160.
[18] Du, X., and Chen, W., 2004, "Sequential Optimization and Reliability Assessment for Probabilistic Design," J. Mech. Des., **126**, pp. 225–233.
[19] Chakrabarti, A., 1998, "Supporting two Views of Function in Mechanical Design, Workshop on Functional Modeling and Teleological Reasoning," 15th AAAI National Conference on Artificial Intelligence, WI.
[20] Chandrasekaran, B., and Josephson, J. R., 2000, "Function in Device Representation," Eng. Comput., **16**(3–4), pp. 162–177.
[21] Pahl, G., Beitz, W., Feldhusen, J., and Grote, K. H., 2007, *Engineering Design: A Systematic Approach*, 3rd ed., Springer Verlag, London.
[22] Otto, K., and Wood, K., *Product Design: Techniques in Reverse Engineering, Systematic Design, and New Product Development* (Prentice-Hall, New York, 2001).
[23] Miles, L., *Techniques of Value Analysis and Engineering* (McGraw-Hill, New York, 1961).
[24] Dieter, G., 1991, *Engineering Design: A Materials and Processing Approach*, 2nd ed., McGraw-Hill, New York.
[25] Ullman, D. G., *The Mechanical Design Process* (McGraw-Hill, New York, 2002).
[26] Cutherell, D., *Chapter 16: Product Architecture* (Wiley, New York, 1996).
[27] Rodenacker, W., *Methodisches Konstruieren (Methodical Design)* (Springer, New York, 1971).
[28] Roth, K., *Konstruieren mit Konstrktionskatalogen (Design with Construction Catalogs)* (Springer Verlag, Berlin, 1982).
[29] Koller, R., *Konstruktionslehre für den Maschinenbau (Mechanical Engineering Design)*, (Springer-Verlag, Berlin, 1985).
[30] Pahl, G., and Beitz, W., 1984, *Engineering Design: A Systematic Approach* Design Council, London.
[31] Hundal, M., 1990, "A Systematic Method for Developing Function Structures, Solutions and Concept Variants," Mech. Mach. Theory, **25**(3), pp. 243–256.
[32] Little, A., Wood, K., and McAdams, D., 1997, "Functional Analysis: A Fundamental Empirical Study for Reverse Engineering, Benchmarking and Redesign," Proceedings of the 1997 Design Engineering Technical Conferences, 97-DETC/DTM-3879, Sacramento, CA.
[33] Szykman, S., Racz, J., and Sriram, R., 1999, "The Representation of Function in Computer-Based Design," ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference, DETC99/DTM-8742, Las Vegas, NV.

[34] Stone, R., and Wood, K., 2000, "Development of a Functional Basis for Design," J. Mech. Des., **122**(4), pp. 359–370.
[35] Hirtz, J., Stone, R., McAdams, D., Szykman, S., and Wood, K., 2002, "A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts," Res. Eng. Des., **13**(2), pp. 65–82.
[36] Chakrabarti, A., and Bligh, T. P., 2001, "A Scheme for Functional Reasoning in Mechanical Conceptual Design," Des. Stud., **22**(6), pp. 493–517.
[37] Chakrabarti, A., 1997, "Deep Understanding and Problem Solving Using Function Structures: A Case Study," Proceedings International Conference in Engineering Design, Tampere, Vol. 3, pp. 71–76.
[38] Andreasen, M. M., 1980, "Syntesemetoder på systemgrundlag," Ph.D. thesis in Lunds Tekniska Högskola.
[39] Freeman, P., and Newell, A, 1971, "A Model for Functional Reasoning in Design," Advanced Papers IJCAI'71, London, pp. 621–640.
[40] Bracewell, R. H., Chaplin, R. V., Langdon, P. M., Li, M., Oh, V. K., Sharpe, J. E. E., and Yan, X. T., "Integrated Platform for AI Support of Complex Design (Part I): Rapid Development of Schemes from First Principles," CACD'95, Springer-Verlag, Lancaster, 1995.
[41] Sturges, R. H., O'Shaughnessy, K. O., and Kilam, M. I., 1996, "Computational Model for Conceptual Design Based on Extended Function Logic," Artif. Intell. Eng. Des. Anal. Manuf., Vol. **10**(4), pp. 225–274.
[42] Chakrabarti A., and Bligh, T. P., 1996, An Approach to Functional Synthesis of Design Concepts: Theory, Application, and Emerging Research Issues, Artif. Intell. Eng. Des. Anal. Manuf., **10**(4), pp. 313–331.
[43] Liu, Y. C., Chakrabarti, A., and Bligh, T. P., 2000, "A Computational Framework for Concept Generation and Exploration in Mechanical Design: Further Developments of FuncSION," *Artificial Intelligence in Design (AID'00)*, J. Gero and F. Sudweeks eds., pp. 499–519.
[44] Gero, J. S., 1990, "Design Prototypes: A Knowledge Representation Schema for Design," AI Mag., **11**(4), pp. 26–36.
[45] Goel, A. K., 1991, "A Model Based Approach to Case Adaptation," *Proceedings 13th Annual Conference of the Cognitive Science Society*, Aug. 1991, Chicago, pp. 143–148.
[46] Y. Umeda, M. Ishii, M. Yoshioka, Y. Shimomura and T. Tomiyama, 1996, "Supporting Conceptual Design Based on the Function–Behaviour–State Modeler," Artif. Intell. Eng. Des. Anal. Manuf., **10**, pp. 275–288.
[47] Deng, Y.-M., Britton, G. A., and Tor, S. B., 2000, "Constraint-Based Functional Design Verification for Conceptual Design," Comput.-Aided Des., **32**, pp. 889–899.
[48] Vargas-Hernandez, N., and Shah, J. J., 2004, "2nd-CAD: A Tool for Conceptual Systems Design in Electromechanical Domain", ASME J. Comput. Inf. Sci. Eng., **4**(1), pp. 28–36.
[49] Chakrabarti, A., Sarkar, P., Leelavathamma, B., and Nataraju, B. S., 2005"A Functional Representation for Aiding Biomimetic and Artificial Inspiration of New Ideas," Artif. Intell. Eng. Des. Anal. Manuf., **19**(2), pp. 113–132.
[50] Erden, M. S., Komoto, H., Van Beek, T. J., D'Amelio, V., Echavarria, E., and Tomiyama, T., 2008, "A Review of Functional Modeling: Approaches and Applications," Artif. Intell. Eng. Des. Anal. Manuf., **22**(2), pp. 147–169.
[51] Chandrasekaran, B., 1994, "Function Representation and Causal Processes," Adv. Comput., **38**, pp. 73–143.
[52] Chakrabarti, A., and Blessing, L., 1996, "Special Issue: Representing Functionality in Design," Artif. Intell. Eng. Des. Anal. Manuf., **10**(4), pp. 251–253.
[53] Winsor, J., and MacCallum, K., 1994, "A Review of Functionality Modelling in Design," Knowl. Eng. Rev., **9**, pp. 163–199.
[54] Stone, R. B., and Chakrabarti, A., 2005, "Special Issue: Engineering Applications of Representations of Function–Part 1," Artif. Intell. Eng. Des. Anal. Manuf., **19**(3), pp. 137–137.
[55] Far, B. H., and Elamy, A. H., "Functional Reasoning Theories: Problems and Perspectives," Artif. Intell. Eng. Des. Anal. Manuf., 19(2), pp. 75–88.
[56] Chakrabarti, A., ed., *Design Synthesis: Issues, Understanding and Methods* (Springer-Verlag, London, 2002).
[57] Falkenhainer, B., Forbus, K., and Gentner, D., 1989, "The Structure-Mapping Engine: Algorithms and Examples," Artif. Intell., **41**(1), pp. 1–63.
[58] Clement, J., 1988, "Observed Methods for Generating Analogies in Scientific Problem Solving," Cogn. Sci., **12**(4), pp. 563–586.
[59] Clement, J., "*Creative Model Construction in Scientists and Students*: *The Role of Imagery, Analogy, and Mental Simulation* (Springer, Dordrecht, 2008).
[60] Christensen, B. T., and Schunn, C. D., 2007, "The Relationship of Analogical Distance to Analogical Function and Pre-Inventive Structure: The Case of Engineering Design," Mem. Cognit., **35**(1), pp. 29–38.
[61] Goel, A., and Bhatta, S., 2004, "Design Patterns: An Unit of Analogical Transfer in Creative Design," Adv. Eng. Inf., **18**(2), pp. 85–94.
[62] Bohm, M., Stone, R., and Szykman, S., 2005, "Enhancing Virtual Product Representations for Advanced Design Repository Systems," J. Comput. Inf. Sci. Eng., **5**(4), pp. 360–372.
[63] Szykman, S., Sriram, R., and Regli, W., 2001, "The Role of Knowledge in Next-generation Product Development Systems," J. Comput. Inf. Sci. Eng., **1**(1), pp. 3–11.
[64] Szykman, S., Fenves, S., Keirouz, W., and Shooter, S., 2001, "A Foundation for Interoperability in Next-Generation Product Development Systems," Comput.-Aided Des., **33**(7), pp. 545–559.
[65] Bohm, M., Stone, R., Simpson, T., and Steva, E., 2008, "Introduction of a Data Schema: To Support a Design Repository," Comput.-Aided Des., **40**(7), pp. 801–811.
[66] Bryant, C., McAdams, D., Stone, R., Kurtoglu, T., and Campbell, M., 2005, "A Computational Technique for Concept Generation," Proceedings of IDETC/CIE 2005, DETC2005-85323, Long Beach, CA.
[67] Bryant, C., Stone, R., McAdams, D., Kurtoglu, T., and Campbell, M., 2005, "Concept Generation from the Functional Basis of Design," International Conference on Engineering Design, ICED 05, Melbourne, Australia.
[68] Shooter, S., Simpson, T., Kumara, S., Stone, R., and Terpenny, J., 2005, "Toward a Multi-Agent Information Management Infrastructure for Product Family Planning and Mass Customisation," Int. J. Mass Customisation, **1**(1), pp. 134–155.
[69] Regli, W., Kopena, J., Grauer, M., Simpson, T., Stone, R., Lewis, K., Bohm, M., Wilkie, D., Piecyk, M., and Osecki, J., 2010, "Archiving the Semantics of Digital Engineering Artifacts: A Case Study," AI Mag., **31**, pp. 37–50.
[70] Bohm, M., Vuchovich, J., and Stone, R., 2007, "An Open Source Application for Archiving Product Design Information," Proceedings of DETC'07, DETC2007-35401, Las Vegas, NV.
[71] Kurtoglu, T., Campbell, M., Bryant, C., Stone, R., and McAdams, D., 2009, "A Component Taxonomy as a Framework for Computational Design Synthesis," ASME J. Comput. Inf. Sci. Eng., **9**(1), p. 011007.
[72] Bohm, M., Vucovich, J., and Stone, R., 2005, "Capturing Creativity: Using a Design Repository to Drive Concept Innovation," Proceedings of IDETC/CIE 2005, DETC2005-85105, Long Beach, CA.
[73] Bohm, M. R., Vucovich, J. P., and Stone, R. B., 2008, "Using a Design Repository to Drive Concept Generation," J. Comput. Inf. Sci. Eng., **8**(1), pp. 014502.
[74] Kurtoglu, T., and Campbell, M., 2008, "Automated Synthesis of Electromechanical Design Configurations from Empirical Analysis of Function to Form Mapping," J. Eng. Des., **20**(1), pp. 83–104.
[75] Kurtoglu, T., Campbell, M., Gonzalez, J., Bryant, C., Stone, R., and McAdams, D., 2005, "Capturing Empirically Derived Design Knowledge for Creating Conceptual Design Configurations," Proceedings of IDETC/CIE 2005, DETC2005-84405, Long Beach, CA.
[76] Chakrabarti, A., Johnson, A. L., and Kiriyama, T., 1997, An Approach to Automated Synthesis of Solution Principles for Micro-Sensor Designs, Proceedings of the International Conference in Engineering Design, Tampere, Vol. 2, pp. 125–128.
[77] Navinchandra, D., Sycara, K. P., and Narasimhan, S., 1991, "A Transformational Approach to Case-Based Synthesis," Artif. Intell. Eng. Des. Anal. Manuf., **5**, pp. 31–45.
[78] Subramanian, D., and Wang, C.-S., 1995, "Kinematic Synthesis with Configuration Spaces," Res. Eng. Des., **7**(3), pp. 193–213.
[79] Williams, B. C., 1990, "Interaction-Based Invention: Designing Novel Devices from First Principles," AAAI-90 Proceedings Eighth National Conference on Artificial Intelligence, Vol. 1, Boston, MA.
[80] Palmer, R. S., and Shapiro, V., 1993, "Chain Models of Physical Behavior for Engineering Analysis and Design," Res. Eng. Des., **5**, pp. 161–184.
[81] Mittal, S., Dym, C., and Morjara, M., 1985, "PRIDE: An Expert System for the Design of Paper Handling Systems," IEEE Comput., **19**(7), pp. 102–114.
[82] Chakrabarti, A., 2001, "Improving Efficiency of Procedures for Computational Synthesis by Using Bidirectional Search," Artif. Intell. Eng. Des. Anal. Manuf., **15**(5), pp. 67–80.
[83] Liu, Y. C., and Chakrabarti, 2010, A. "Investigation of Design Heuristics for Pruning the Number of Mechanism Solutions in Computer-Based Conceptual Design," 2nd International Conference on Computer and Automation Engineering (ICCAE 2010), Feb. 26–28, Singapore.
[84] Chakrabarti, A., and Johnson, A. L., 1999, "Detecting Side Effects in Solution Principles," *Procedings of the International Conference on Engineering Design (ICED99)*, Munich, Vol. 2, pp. 661–666.
[85] Langdon, P., and Chakrabarti, A., 1999, "Browsing a Large Solution Space in Breadth and Depth," *Proceedings of the International Conference on Engineering Design (ICED99)*, Munich, Vol. 3, pp. 1865–1868.
[86] Ulrich, K. T., and Seering, W. P., 1988, "Function Sharing in Mechanical Design," *AAAI-88 Proceedings*, pp. 342–346.
[87] Chakrabarti, A., 2004, "A New Approach to Structure Sharing," ASME J. Comput. Inf. Sci. Eng., **1**(1), pp. 1–78.
[88] Fricke, G., 1996, "Successful Individual Approaches in Engineering Design," Res. Eng. Des. **8**, pp. 151–165.
[89] Liu, Y-C., Chakrabarti, A., and Bligh, T. P., 2003, "Towards an Ideal Approach for Concept Generation," Des. Stud., **24**(4), pp. 341–355.
[90] Bryant, C., Pieper, E., Walther, B., Kurtoglu, T., Stone, R., McAdams, D., and Campbell, M., 2006, "Software Evaluation of an Automated Concept Generator Design Tool," Proceedings of the 2006 ASEE Annual Conference, ASEE-2006-1758, Chicago, IL.
[91] Bryant, C., McAdams, D., Stone, R., Kurtoglu, T., and Campbell, M., 2006, "A Validation Study of an Automated Concept Generator Design Tool," Proceedings of IDETC/CIE 2006, DETC2006-99489, Philadelphia, PA.
[92] Kurtoglu, T., and Campbell, M., 2007, "Exploring the Worth of Automatically Generated Design Alternatives Based on Designer Preferences," International Conference on Engineering Design, Paris, France.
[93] Gips, J., and Stiny, G., 1980, "Production Systems and Grammars: A Uniform Characterization," Environ. Plann. B, **7**, pp. 399–408.
[94] Krishnamurti, R., and Stouffs, R., 1993, Spatial Grammars: Motivation, Comparison, and New Results, *5th International Conference on Computer-Aided Architectural Design Futures*, North-Holland Publishing Co., Pittsburgh, pp. 57–74.
[95] Antonsson, E. K., and Cagan, J., 2001, *Formal Engineering Design Synthesis*, Cambridge University, Cambridge, England, pp. 65–91
[96] Nagl, M., 1976, "Formal Languages of Labeled Graphs," J. Comput., **16**(1–2), pp. 113–137.
[97] Chomsky, N., 1980, *Studies on Semantics in Generative Grammar* (Walter de Gruyter, New York, NY, 1980).

[98] Ehrig, H., Kreowski, H. J., Maggiolo-Schettini, A., Rosen, B. K., and Winkowski, J., 1981, "Transformations of Structures—An Algebraic Approach," J Math. Syst. Theory, **14**(4), pp. 305–334.
[99] Geiß, R., Batz, G. V., Grund, D., Hack, S., and Szalkowski, A., 2006, "GrGen: A Fast SPO-Based Graph Rewriting Tool," *Graph Transformations*, Springer, Berlin, pp. 383–397.
[100] Helms, B., and Shea, K., 2010, "Object-Oriented Concepts for Computational Design Synthesis," *11th International Design Conference DESIGN 2010*,D. Marjanović, M. Štorga, N. Pavković, and N. Bojčetić, eds.,. Dubrovnik, Croatia.
[101] Helms, B., Shea, K., and Hoisl, F., 2009, "A Framework for Computational Design Synthesis Based on Graph-Grammars and Function-Behavior-Structure," ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. San Diego.
[102] Alber, R., and Rudolph, S., 2003, "'43'—A Generic Approach for Engineering Design Grammars," Proceedings AAAI Spring Symposium "Computational Synthesis," Stanford, CA, AAAI Technical Report SS-03-02.
[103] Schäfer, J., and Rudolph, S., 2005, "Satellite Design by Design Grammars," Aerosp. Sci. Technol., **9**(1), pp. 81–91.
[104] Patel, J., and Campbell M. I., 2010, "An Approach to Automate and Optimize Concept Generation of Sheet Metal Parts by Topological and Parametric Decoupling," ASME J. Mech. Des.," **132**(5), p. 051001.
[105] Agu, D., and Campbell, M. I., 2010, "Automated Analysis of Product Disassembly to Determine Environmental Impact," Int. J. Sustain. Des., **1**, pp. 241–256.
[106] Brimble, R., and Sellini F., 2000, The MOKA Modelling Language, Knowledge Engineering and Knowledge Management Methods, Models, and Tools (Lecture Notes in Computer Science) Vol. 1937/2000, pp. 95–120.
[107] Kerzhner, A. A., and Paredis C. J. J., "Using Domain Specific Languages to Capture Design Synthesis Knowledge for Model-Based Systems Engineering," ASME 2009 International Design Engineering Technical Conference and Computers and Information in Engineering Conference IDETC/CIE2009, San Diego, DETC2009-87286.
[108] Chau, H. H., Chen, X. J., McKay, A., and de Pennington, A., 2004, "Evaluation of a 3D Shape Grammar Implementation," *Design Computing and Cognition 04*, J. S. Gero, ed., Kluwer Academic Publishers, Cambridge, pp. 357–376.
[109] Hoisl, F., and Shea, K., 2011, "An Interactive, Visual Approach to Developing and Applying Parametric 3D Spatial Grammars", Artif. Intell. Eng. Des. Anal. Manuf., **25**(4), in press.
[110] McCormack, J. P., and Cagan, J., 2002, "Supporting Designer's Hierarchies through Parametric Shape Recognition," Environ. Plan. B: Plan. Des., **29**, pp. 913–931.
[111] McCormack, J. P., and J. Cagan, 2006, "Curve-Based Shape Matching – Supporting Designers' Hierarchies Through Parametric Shape Recognition of Arbitrary Geometry," Environ. Plann. B, **33**(4), pp. 523–540.
[112] Jowers, I., Hogg, D., McKay, A., Chau, H., and de Pennington, A., 2010, "Shape Detection With Vision: Implementing Shape Grammars in Conceptual Design," Res. Eng. Des., **21**, pp. 235–247.
[113] Heisserman, J., Mattikalli, R., and Callahan, S., 2004, "A Grammatical Approach to Design Generation and its Application to Aircraft Systems," Proceedings of Generative CAD Systems Symposium '04, Pittsburgh, Pennsylvania, July 12–14, 2004.
[114] Heisserman, J., 1994, "Generative Geometric Design," IEEE Comput. Graphics Appl., **14**(2), pp. 37–45.
[115] Yogev, O., Shapiro, A. A., and Antonsson, E. K., 2009, "Computational Evolutionary Embryogeny," IEEE Trans. Evol. Comput., **14**(2), pp. 301–325
[116] Orsborn, S., Boatwright, P., and Cagan, J., 2008, "Identifying Product Shape Relationships Using Principal Component Analysis," Res. Eng. Des., **18**(4), pp. 181–196.
[117] Orsborn, S., Cagan, J., and Boatwright, P., 2008, "A Methodology for Creating a Statistically Derived Shape Grammar Composed of Non-Obvious Shape Chunks," Res. Eng. Des., **18**(4), pp. 163–180.
[118] Knight, T. W., 2000, "Shape Grammars in Education and Practice: History and Prospects," Int. J. Des. Comput., **2**.
[119] Gick, M. L., and Holyoak, K. J., 1980, "Analogical Problem Solving," Cogn. Psychol., **12**, pp. 306–355.
[120] Gordon, W. J. J., *Synectics* (Harper & Row, NY, 1961)
[121] Goel, A. K., 1997, "Design, Analogy and Creativity," IEEE Expert Intell. Syst. Appl., **12**, pp. 62–70.
[122] Gentner, D., 1983, "Structure-Mapping: A Theoretical Framework for Analogy," Cogn. Sci., **7**(2), pp. 155–170.
[123] Gick, M., and Holyoak, K. J., 1983, "Schema Induction and Analogical Transfer," Cogn. Psychol., **15**(1), pp. 1–38.
[124] Bhatta, S., and Goel, A., 1997, "Learning Generic Mechanisms for Innovative Design Adaptation," J. Learn. Sci., **6**(4), pp. 367–396.
[125] Gero, J. S., and Maher, M. L., 1991, "Mutation and Analogy to Support Creativity in Computer-aided design," *Proceedings of the Computer Aided Architectural Design Futures '91*, Zurich, pp. 241–249.
[126] Davies, J., and Goel, A. K., 2001, "Visual Analogy in Problem Solving," Proceedings of the International Joint Conference on Artificial Intelligence.
[127] Tseng, I., Moss, J., Cagan, J., and Kotovsky, K., 2008, "The Role of Timing and Analogical Similarity in the Stimulation of Idea Generation in Design," Des. Stud., **29**, pp. 203–221.
[128] McAdams, D. A., and Wood, K. L., 2002, "A Quantitative Similarity Metric for Design by Analogy," J. Mech. Des., **124**(2), pp. 173–182.
[129] Wiratunga, N., Craw, S. and Rowe, R., 2002, "Learning to Adapt for Case-Based Design," *Proceedings of the Sixth European Conference on Case-Based Reasoning*, Springer-Verlag, Aberdeen, Scotland, Sept. 4–7, pp. 421–435.
[130] Fantoni, G., Taviani, C., and Santoro, R., 2007, "Design by Functional Synonyms and Antonyms: A Structured Creative Technique Based on Functional Analysis," Proc. Inst. Mech. Eng., Part B, **221**, pp. 673–683.
[131] Linsey, J., Wood, K., and Markman, A., 2008, "WordTrees: A Method for Design-by-Analogy," Proceedings of the 2008 ASEE Annual Conference and Exhibition, Pittsburgh, PA.
[132] Bohm, M. R., and Stone, R. B., 2009, A Natural Language To Component Term Methodology: Towards A Form Based Concept Generation Tool, DETC2009-86581.
[133] Yamamoto, E., Taura, T., and Ohashi, S., 2009. "Thesaurus for Natural-Language-Based Conceptual Design," The ASME 2009 International Design Engineering Technical Conference and Computers and Information in Engineering Conference IDETC/CIE2009, San Diego, DETC2009-86943.
[134] Balazs, M. E., and Brown, D.C., "Design Simplification by Analogical Reasoning," *Knowledge Intensive Computer Aided Design* (Kluwer Academic, Dordrecht, Netherlands, 2001).
[135] Börner, K., Pippig, E., Tammer, E., and Coulonet, C., 1996, "Structural Similarity and Adaptation," *Proceedings of the Third European Workshop Case-Based Reasoning*, Springer-Verlag, New York, pp. 58–75.
[136] Qian, L., and Gero, J., 1992 "A Design Support System Using Analogy," *Proceedings of the Second International Conference AI in Design*, Kluwer Academic Publishers, Dordrecht, The Netherlands, pp. 795–813.
[137] Bhatta, S., Goel, A. K., and Prabhakar, S., 1994, "Innovation in Analogical Design: A Model-Based Approach," *Proceedings of Third International Conference AI in Design*, Kluwer, pp. 57–74.
[138] Kulinski, J., and Gero, J. S., 2001, "Constructive Representation in Situated Analogy in Design," *CAADFutures 2001*, B. de Vries, J. van Leeuwen, and H. Achten, eds., Kluwer, Dordrecht, pp. 507–520.
[139] Kolodner, J. L., *Case-Based Reasoning* (Morgan Kaufmann, California, 1993).
[140] Watson, I., and Perera R. S., 1997, Case Based Design: A Review and Analysis of Building Design Applications, Artif. Intell. Eng. Des. Anal. Manuf., **11**(1), pp. 59–87.
[141] Reisbeck, C. K., and Schank, R. C., *Inside Case-Based Reasoning* (Lawrence Erlbaum Associates, Hillsdale, NJ, 1989).
[142] Aamodt, A., and Plaza, E., 1994, "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches," AI Commun., **7**(1), pp. 39–59.
[143] Heylighen, A., and Neuckermans, H., 2001, A Case Base for Case Based Design Tools for Architecture," Comput.- Aided Des., **33**, pp. 1111–1122.
[144] Goel, A. K., and Craw, S., 2006, "Design, Innovation and Case-Based Reasoning," Knowl. Eng. Rev., **20**(3), pp. 271–276.
[145] Watson, I., and Marir, F., 1994, "Case-Based Reasoning: A review," Knowl. Eng. Rev., **9**(4), pp. 327–354.
[146] Chandrasekaran, B., 1990, "Design Problem Solving: A Task Analysis," AI Mag. **11**, pp. 59–71.
[147] Schank, R. C., and Abelson, R. P., *Scripts, Plans, Goals and Understanding* (Erlbaum, Hillsdale, New Jersey, 1977).
[148] Kolodner, J. L., 1983, "Maintaining Organization in a Dynamic Long-Term Memory," Cognit. Sci., **7**(4), pp. 243–280.
[149] Kolodner, J. L., 1983, "Reconstructive Memory: A Computer Model," Cognit. Sci., **7**(4), pp. 281–328.
[150] Navinchandra, D., 1987, "Exploring for Innovative Designs by Relaxing Criteria and Reasoning from Precedent-Based Knowledge," Ph.D. dissertation, M.I.T.
[151] Maher, M. L., and Zhao, F., 1987, "Using Experience to Plan the Synthesis of New Designs," *Expert Systems in Computer Aided Design*, J. Gero, ed., North Holland, Amsterdam, The Netherlands.
[152] Schank, R., *Dynamic Memory: A Theory of Reminding and Learning in Computers and People* (Cambridge University, Cambridge, UK, 1982)
[153] Porter, B. W., and Bareiss, E. R., 1986, "PROTOS: An Experiment in Knowledge Acquisition for Heuristic Classification Tasks," *Proceedings of the First International Meeting on Advances in Learning (IMAL)*, Les Arcs, France, pp. 159–174.
[154] Falkeneheimer, B., Forbus, K. D., and Gentner, D., 1986, "The Structure Mapping Engine," Proceeding of the Sixth National Conference on Artificial Intelligence, Philadelphia, PA.
[155] Navinchandra, D., *Exploration and Innovation in Design: Towards a Computational Model* (Springer Verlag, New York, NY, 1991).
[156] Maher, M. L., and Zhang, D. M., 1991, "CADSYN: Using Case and Decomposition Knowledge for Design Synthesis," *Artificial Intelligence in Design*, J. S. Gero, ed., Butterworth-Heinmann, Oxford, UK.
[157] Domeshek, E., 1993, "A Case Study of Case Indexing: Designing Index Feature Sets to Suit Task Demands and Support Parallelism," *Advances in Connectionnist and Neural Computation Theory*, *Analogical connections* J. Barenden and K. Holyoak, eds., Vol. 2, Norwood, NJ.
[158] Avramenko, Y., and Kraslawski A., 2008, *Case Based Design: Applications in Process Engineering, Studies in Computational Intelligence*, Springer-Verlag, Berlin, Vol. 87.
[159] Chakrabarti, A., and Shu, L. H., 2010, "Guest Editorial: Biologically Inspired Design," Artif. Intell. Eng. Des. Anal. Manuf., **24**, pp. 453–454.
[160] Benami, O., and Jin, Y., 2002, "Creative Stimulation in Conceptual Design," Proceedings of ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference (DETC/CIE), Montreal, QC, Canada, DETC2002/DTM-34023.
[161] Helms, M. E., Vattam, S. S., and Goel, A. K., 2009, "Biologically Inspired Design: Process and Products," Des. Stud., **30**(5), pp. 606–622.
[162] Vattam, S. S., Helms, M. E., and Goel, A. K., 2008, "Compound Analogical Design: Interaction Between Problem Decomposition and Analogical Transfer in

Biologically Inspired Design," *Proceedings of the Third International Conference on Design Computing and Cognition*, Atlanta, June, Springer, Berlin, pp. 377–396.
[163] Vattam, S. S., Helms, M. E., and Goel, A. K., 2010, "A Content Account of Creative Analogies in Biologically Inspired Design," Artif. Intell. Eng. Des. Anal. Manuf., **24**, pp. 467–481.
[164] Sartori, J., Pal, U., and Chakrabarti, A., 2010, "A Methodology for Supporting "Transfer" in Biomimetic Design," Artif. Intell. Eng. Des. Anal. Manuf., **24**, pp. 483–505.
[165] Hill, B., *Innovationsquelle Natur: Naturorientierte Innovationsstrategie für Entwickler, Konstrukteure und Designer* (Shaker, Aachen, Germany, 1997).
[166] Hill, B., 2005, "Goal Setting through Contradiction Analysis in the Bionics-Oriented Construction Process," CIM, Blackwell Publishing Ltd, Oxford, Vol. **14**(19), pp. 59–65.
[167] Schild, K., Herstatt, C., and Lüthje, C., *How to Use Analogies for Breakthrough Innovations*, (Technical University of Hamburg, Institute of Technology and Innovation Management, Hamburg, 2004).
[168] Gramann, J., 2004, "Problemmodelle und Bionik als Methode," Dissertation, TU, Munich.
[169] Vincent, J., and Mann, D., 2002, "Systematic Technology Transfer From Biology to Engineering," Philos. Trans. R. Soc. London, **360**, pp. 159–173.
[170] Vincent, J. F. V., Bogatyreva, O. A., Bogatyrev, N. R., Bowyer, A., and Pahl, A. K., 2006, "Biomimetics: Its Practice and Theory," J. R. Soc., Interface, **3**, pp. 471–482.
[171] Tinsley, A., Midha, P., Nagel, R., McAdams, D., Stone, R., and Shu, L., 2007, "Exploring the Use of Functional Models as a Foundation for Biomimetic Conceptual Design," ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Las Vegas, Nevada, DETC 2007-35604.
[172] Vattam, S., Wiltgen, B., Helms, M., Goel, A. K., and Yen, J., 2010, "DANE: Fostering Creativity in and through Biologically Inspired Design," To appear in Proceedings of 1st International Conference on Design Creativity (ICDC2010), December, Kobe, Japan.
[173] Wilson, J., Chang, P., Yim, S., and Rosen, D., 2009, "Developing a Bio-Inspired Design Repository Using Ontologies," Proceedings of ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference, IDETC/CIE, San Diego, CA, DETC2009-87272.
[174] Wilson, J. O., and Rosen, D., 2007, "Systematic Reverse Engineering of Biological Systems," Proceedings of ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE), Las Vegas, NV, DETC2007/DTM-35395.
[175] Hacco, E., and Shu, L., 2002, "Biomimetic Concept Generation Applied to Design for Remanufacture," Proceedings of ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Montreal, Sept. 29–Oct. 2, DETC2002/DFM-34177.
[176] Shu, L. H., 2010, "A Natural-Language Approach to Biomimetic Design," Artif. Intell. Eng. Des., Anal. Manuf., **24**, pp. 507–519.
[177] Sarkar, P., Phaneendra, S., and Chakrabarti, A., 2008, "Developing Engineering Products Using Inspiration From Nature," ASME J. Inf. Sci. Eng., **8**(3), pp. 031001.
[178] Srinivasan, V., and Chakrabarti, A., Supporting Process and Product Knowledge in Biomimetic Design, Special Issue on Design and Nature, I. C. Gebeshuber, H. Abdel-Aal, Guest Editors, Int. J. Des. Eng., Inderscience (In press).
[179] Boyer, F., Chablat, D., Lemoine, P., and Wenger, P., 2009, "The Eel-Like Robot," Proceedings of ASME Design Engineering Technical Conference and Computers and Information in Engineering Conference, Sand Diego, CA, DETC2009-86328.
[180] Chakrabarti, A., Ojha, S., Pal, U., Ranjan, B. S. C., Srinivasan, V., and Ranganath, R., 2009, "Exploring Serially Connected Multi-Tracked All-Terrain Vehicles for Improved Obstacle Climbing Performance," 14th National Conference on Machines and Mechanisms (NaCoMM09), Durgapur.