CIRP Journal of Manufacturing Science and Technology 1 (2009) 120–125

Contents lists available at ScienceDirect

CIRP Journal of Manufacturing Science and Technology

journal homepage: www.elsevier.com/locate/cirpj

Structure and models of artifactual routine design problems for computational synthesis

J.M. Jauregui-Becker *, H. Tragter, F.J.A.M. van Houten

Laboratory of Design Production and Management, Faculty of Engineering Technology, University of Twente, Enschede, The Netherlands

ARTICLE INFO

Article history:
Available online 20 November 2008

Keywords:
Computational synthesis
Artifactual routine design
Problem structure
Problem models

ABSTRACT

Computational synthesis (CS) researches the automatic generation of solutions to design problems. The aim is to shorten design times and present the user with multiple design solutions. However, initializing a new CS process has not received much attention in literature. With this motivation, this paper presents a framework to structure and model routine design to assist the development of new CS processes. First, concepts are presented and used to propose a structure for artifactual routine design problems. Latter, base models (building blocks) for creating new designs are described. Finally, a classification of design families according to its structure and models is presented together with its relation to know CS methods.

1. Introduction

Research in computational synthesis (CS) studies algorithmic procedures to automate the generation of designs. This is done by combining "low-level" building blocks in such a manner that "high level" functionalities can be achieved. CS methods vary from, among others, straight forward implementation of artificial-intelligence, constraint solving and optimization techniques down to much more specialized approaches as in shape grammars [1] and A-design [2].

A well-accepted model for CS is shown in Fig. 1, as presented in [3]. The model highlights the processes a CS system should resemble in order to automatically generate designs solutions. In the flowchart, the design problem is first formulated by the user, which (in engineering design problems) is usually done by declaring variables, constraints, and constructing objective functions. This information is then assembled into *representations* (models) that can support the computational processes that take care for the *generation* of candidate solutions. A candidate solution is one that satisfies all constraints in the problem, independently on how well the goal is achieved. An *evaluation* step analyses the results by calculating its performances and decides whether to accept, adjust or reject a solution. *Guidance* drives the generation process in a given direction, with the goal of generating improved solutions.

As representations define the level of detail and focus of the computational search processes, one can argue that it is the key challenge in initializing a CS process. Furthermore, representations have to model and reassemble the structure of the problem such that the appropriate generation and search mechanism can be determined. However, and as indicated by Cagan et al. [3], the act of initializing a CS process has not received much attention in literature, as most computational synthesis methods are developed to solve a particular design problem. To cope with this, this paper presents a framework to structure and model design problem formulations. While structuring a problem permits defining strategies for automating the CS process, modeling it permits describing the building blocks and algorithms to both represent the problem and generate solutions. Strategies consist of procedures to decompose a problem and later integrate generated solutions. Algorithms take care about instantiating the variables and are therefore more dependent on the type models used in the representation. The authors believe that counting with such a framework assists the:

- identification of types of design problems based on its structures rather than on its specific context sensitive information,
- development of CS strategies based on problem structure, and selection (or development) of CS algorithms based on the problem model, and
- future development of reasoning engines that automate the identification of problem structures and take decisions about strategies and algorithms to automate the CS process.

* Corresponding author at: Laboratory of Design Production and Management, Faculty of Engineering Technology, University of Twente, P.O. Box 217, Enschede, The Netherlands. Tel.: +31 53 489 2266; fax: +31 53 489 3631.
E-mail address: j.m.jaureguibecker@ctw.utwente.nl (J.M. Jauregui-Becker).

1755-5817/$ – see front matter © 2008 CIRP.
doi:10.1016/j.cirpj.2008.10.002

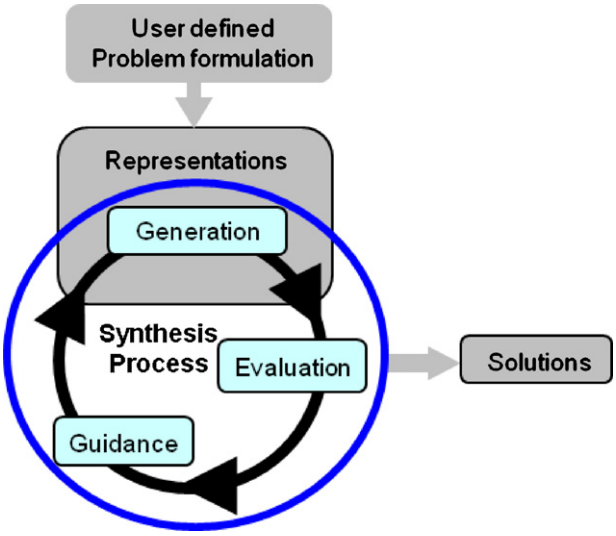


Fig. 1. Computational synthesis model [3].

This paper is organized as follows: Section 2 presents an descriptive analysis for deriving a framework to structure routine design problems; Section 3 introduces models for artifactual representation and generation; Section 4 uses the structure and models to categorize four common types of design problems; and finally, Section 5 presents a summary and follow-up research to this paper.

2. Structure of routine design problems

The structure of design problems has been an important subject of research in the field of Problem Solving Theory (PST) [4–6] as well as in the field of CS [3,7]. From a PST perspective, Simon [4] has defined problem structuring as the process of drawing upon our knowledge to compensate for missing information, and using this knowledge to construct the problem space. From here, it follows that design problems whose problem space are completely defined are regarded as well-defined, while those which are not are regarded as ill-defined. Characteristic to well-defined problems is that they can be structured as function of a known initial state, a clear goal state, a constrained set of logical states and constraint parameters [5]. Additionally, well-defined problems can be solved using generally applicable problem-solving mechanisms, whereas ill-defined problems require more creative approaches [6].

As routine design problems proceed within a known space of functions, expected behaviors and structure variables and the problem is one of instantiating structure variables [7], it is regarded as well defined. Furthermore, the space of designs produced is substantially smaller than the space of possible designs, given that the ranges of applicable values for variables are constrained. Therefore, in this paper a routine design problem is structured when:

- its parts and interrelations are formalized according to their role in the process of creating new designs and
- it meets the conditions of well-defined problems, as in PST.

Sections 2.1–2.5 present definitions, which are used in Section 2.6 to introduce a framework to structure routine design problems. Section 2.7 proposes Semantic Networks as means to represent problem structures.

2.1. Artifacts descriptions

Design artifacts can be described by three different types of entities: (a) vocabulary of elements, (b) descriptions of elements and (c) the configuration of elements. Consider the case of the gear device shown in Fig. 2. Here, the vocabulary of elements is two gears and two shafts. Descriptions determine the attributes of the artifact, like the diameter (D) and angular velocity (w). Configurations determine the disposition of the elements in the structure, as for example the connectedness between elements represented by the relations in the figure. Configurations can be classified into topologic relations and physical coherence constraints. While the former define the topology of the elements in the structure, the latter is used to assure no physical impossibilities are committed by the artifact being designed. For example, two gears cannot share the same place in space. Furthermore, when formulating design problems, only relevant descriptions need to be taken into account. Consider the gear device design, if the designer is only interested in determining the gears diameters to deliver a given angular velocity, no further variables need be taken into account. In [8], a method is presented to aid the identification of the information required for formulating a given routine design problem. The methodology uses FBS modeling to assess the Function, Behavior, Principle, State and Structure representations of the artifact in question.

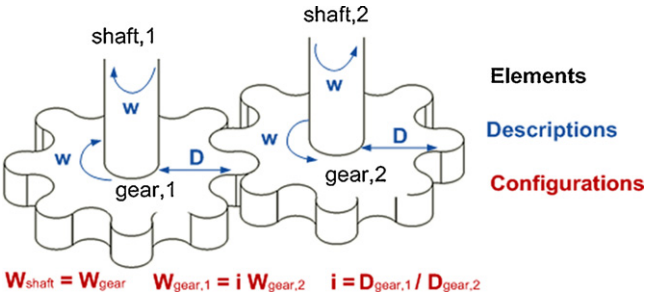


Fig. 2. Gear device: elements, descriptions and configurations.

2.2. Embodiment

Embodiment is here defined as the subset of representations of an artifact upon which instances are created to generate design solutions. In Fig. 3 this is shown for the case of the gear device, where several descriptions are required to model the whole design artifact. However, since the purpose of design is limited, only the diameters and velocity are considered. The embodiment in this case is composed of two elements: one input gear and one output gear. Descriptions are its diameter and velocity.

2.3. Scenario

Artifacts exist in the natural world, and therefore are exposed to environments. An artifact's ability to accomplish its function is greatly affected by its interaction with its environment. The subset of environment variables, attributed to elements in the natural world and considered in measuring a design artifact's ability to accomplish its function, is here defined as scenario. Consider the case of the gear device design in Fig. 3. Scenario can be a shaft attached to the input gear and one connected to the output gear. As for embodiments, descriptions are used to specify scenarios. For example, the rotational speed of the shafts.

2.4. Design goals

Given that design functions are expressed in abstract terms, it is necessary to use measurable descriptions to, both, express and assess its goals. Goals are commonly represented by objective functions, where performance indicators are weighted and added

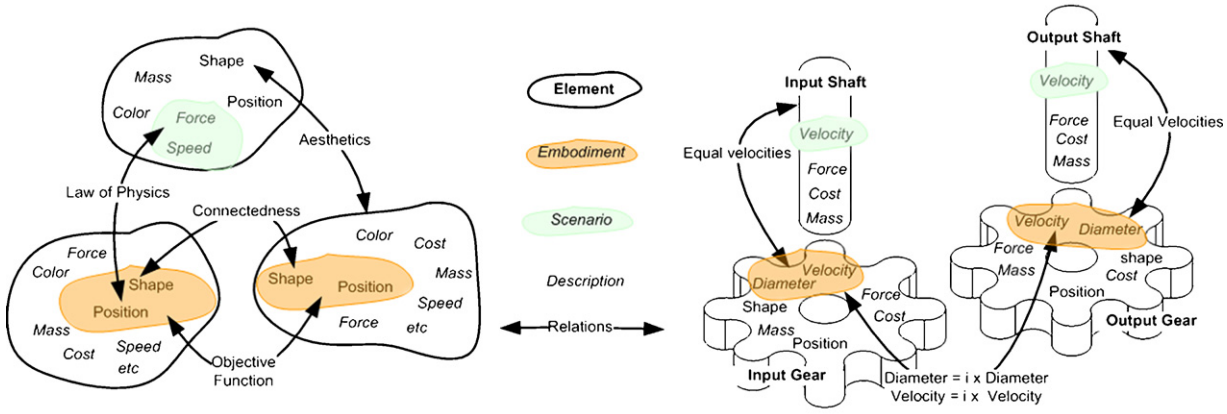


Fig. 3. Gear device: elements, descriptions and configurations.

to compute the overall performance of the design. Performances are calculated by analysis relations using instantiated embodiments and scenarios. Analysis relations use known principles – as for example laws of physics and economics – to model the interaction of the design artifact with its environment and predict its behavior.

Goals can be *expressed* by defining requirements on the objectives functions and performance parameters. On the other hand, goals are *assessed* by calculating – using analysis relation – the performances and objective function of an instantiated artifact. Furthermore, two types of goals can be found in a design problem, namely, constraint satisfaction and optimization. In constraint satisfaction, the objective is finding instances of design artifacts within the allowed topology relations and confinement constraints. Here, performances are used as means of assessing the design. For optimization, maximization or minimization of performances is added to the constraint satisfaction problem. Performances are used to express the desired quality of the design artifact.

Analysis in design is often done analytically, by simulation, or by a combination of both. They can vary from algebraic equations to complex differential equations. Finite element methods, computational fluid dynamics, circuit simulation, and other computational analysis tools offer accurate and robust analyses [3].

For the case of the gear device in Fig. 3, a performance indicator could be the rotational speed of the output shaft. The objective function could be expressed by this performance, reducing the goal to that of the output rotational speed. The goal can then be expressed by a required rotational speed, while it can be assessed by an analysis technique for an instantiated design.

2.5. Synthesis knowledge

Synthesis knowledge determines values of the embodiment as function of scenario and performances descriptions. Here, synthesis knowledge is considered independent from the CS strategies and algorithms. Synthesis knowledge is often the result of experience, and lowers design complexity by further constraining the ranges of permitted value of the embodiment. Synthesis knowledge can also aid the generation, evaluation and guidance processes by providing shortcuts to values that have proven to be successful in design practice.

2.6. Structuring framework

Applying the concepts described before, in this paper a routine design problem is structured at two different abstractions: a *problem class* and a *problem instance*.

A *problem class* is structured in:

- Elements: are considered class descriptions, and are used to represent both, embodiment and scenario elements. Element's responding to functions that cannot coexist simultaneously are structured separately.
- Relations: are considered class descriptions and are of different types, namely: topology, physical coherence, analysis and objective. Their descriptions can be declared within the scope of the class or referred by pointing towards descriptions of embodiment and scenario elements.
- Descriptions: are variables that characterize elements and relations by mathematic models (see Section 3).

A *problem instance* is structured by instantiating scenario and performances descriptions, becoming scenario specifications and performance requirements, respectively. Furthermore, a partiality of embodiment descriptions might also be instantiated, imposing constraints to the space of possible design solutions. These are regarded as embodiment requirements. One problem-instance might have several design solutions, each presented as a different solution instance of the same problem instance. This obliges solution generation algorithms and software architecture to be capable of creating solutions independent from where embodiment requirements, scenario specifications and performance requirements are set. By following this structure, CS strategies and algorithms have to be valid for a design problem class, rather than for specific problem instance.

Using this structure results in a problem formulation where:

- *embodiment elements* and *scenario elements* describe the initial state of the design,
- *objective function*, *performance indicators* and *analysis relations* express and assess the goal of the design artifact,
- *topology relations* and *physical coherence* constraints indicate the set of logical states that have to hold for the design artifact to exist,
- *confinement constraints* restrict the values that embodiment, scenario and performance descriptions are allowed to reach;

which satisfies the conditions of a well-defined problem, as stated in Section 2.

2.7. Design structure representation

Designing requires representations with sufficient expressive power to capture the nature of the concepts while supporting

design processes [7]. In the case concerning this paper, they also have to allow designing the computational processes that will generate design solutions. In this research, a Semantic Networks [9] based representation has been chosen. Semantic Networks are attributed graphs, where the nodes represent concepts, arcs represent relations between concepts, and labels are used to represent properties of both nodes and arcs. The following guidelines are used to represent the problem structure:

- *Nodes*: represent elements. Have a type, which depends on whether the element belongs to the embodiment or the scenario.
- *Arcs*: represent the relation among the elements. Relations also have a type, which depends on whether the relation is topologic, coherence, analysis or objective function.
- *Node labels*: uses descriptions and its confinement constraints to elaborate on the class definition of the element.
- *Arc labels*: specifies the model of the relation by relating element descriptions and independent descriptions if required. A weight factor is an example of an independent description.

In Fig. 4, a semantic network is presented to represent the design of the gear device in Fig. 2. Here, two embodiment elements are shown, namely, an input gear and an output gear. Two scenario elements (input shaft and output shaft) model the relation between the embodiment and its environment. Each node is provided by a label containing the descriptions that are considered in the design problem. The elements are interconnected by three relations, describing the topology of the elements in the artifact. The example is meant to illustrate the representation, and therefore omits analysis relations, coherence constraints and objective function.

3. Models for artifact representation

So far, a scheme for to structure a routine design problem has been presented. This section presents models to represent descriptions and relations. The aim is to further structure the problem by using common data models, which can be regarded as basic building blocks for formulating artifact design problems. Doing so facilitates the development of abstract formulations from where to differentiate families of design problems, and later develop algorithms to automate them.

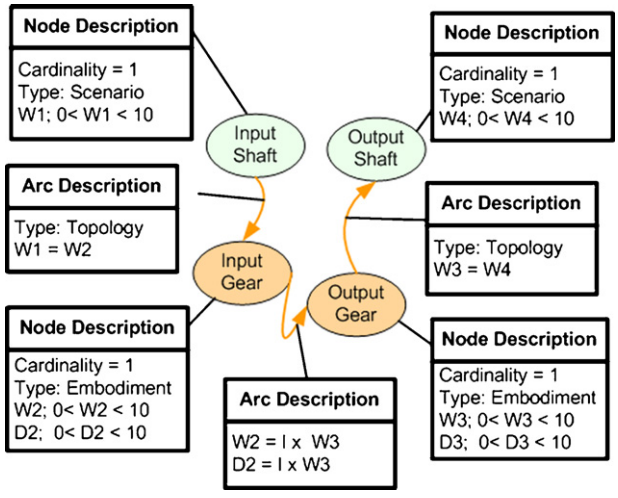


Fig. 4. Gear device design problem representation.

3.1. Descriptions

Descriptions are classified in five model categories: parameter, field, space, shape and topology. Each category represents a complexity dimension in the problem space, as different problem solving approaches are required to generate solutions (see Section 4). Design problems formulated as a function of more than one dimension have a higher degree of complexity and require different methods to automate the generation of solutions.

3.1.1. Parameter

Parameters model properties valid for the entire element. They are used to represent attributes as material properties, color, weight, density, etc. These can be of different nature, as for example numeric, symbolic, logic, predicate, and combinations among them. For numeric parameters, confinement constraints define a continuous or discrete space of possible values. For symbolic, predicate and logic ones, all possible values have to be specified. That is for every $A = \{A_1 \ldots A_n\}$, exists an A_i such that:

$$A_i \in (N, Z, Q, R, C), \quad \text{or } A_i = \{\text{true}, \text{false}\}, \quad \text{or } A_i = \text{Symbol}.$$

3.1.2. Space

Describe positional attributes of the elements in a topology. Positional descriptions depend on the chosen coordinate system – Cartesian, cylindrical of spherical – and the dimensions of interest – 1D, 2D, or 3D. These can be considered as numeric parameters related by a model that is determined by the chosen coordinate system. Values can either be continuous or discrete.

3.1.3. Field

Use parameters and geometric vectors to describe properties that are valid in specific regions of the elements. Fields are specified together with an incident zone, shown in Fig. 5 as cubic.

Incident zone is the spatial place where the field influences an element. An incident zone can be a volume, an area, a line or a point. Meshed CAD models are used in Computer Aided Engineering (CAE) software to specify incident zones. Vectors and parameters can then be attributed to each mesh-element. In Fig. 5 an example illustrates how a parameters and vectors are related with their incident zones.

3.1.4. Shape

Describe the form of the elements – or groups of elements – present in the design structure. Commonly used models are based on geometric relations and graphs.

Geometry models the form by means of mathematic equations. Models are defined with parameters related by geometric

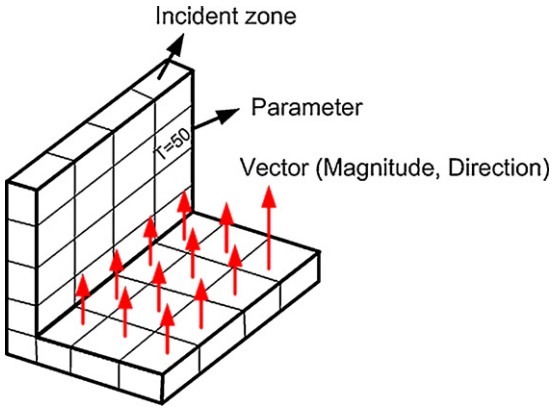


Fig. 5. Example of a field attribute.

functions, the latest being usually algebraic or differential. When the geometric function is known, parameters are instantiated to produce new shapes. When not known, the problem becomes one of finding the correct geometric relations. Polynomials are often used for this purpose, having the polynomial degree and its coefficients as unknowns. Depending on the case, confinement constraints are set either to parameters or geometric functions. Furthermore, perimeters, areas and volumes can also be subject of restriction. Super quadrics [10] have been broadly used for this purpose. Super quadrics are polynomials whose degrees and coefficients values vary depending on the shape being modeled. In Fig. 6 a toroid super quadric shape is shown together with its polynomial equation. By changing the values of the polynomial coefficient, the shape of the toroid can be changed.

Attributed graphs model shapes using nodes (representing shape primitives) and arcs (representing the connection within primitives). Primitives might be further decomposed into subgraphs, obtaining shape models with several levels of abstraction. When the shape graph is specified, design generation consist in instantiating attributes related to the graph's nodes and arcs. When the graph is not specified, but its primitives and relations are, shapes are generated by constructing new graphs. Confinement constraints can be set to both, the number and the types of primitives and relations. In Fig. 7, a handmade symbol of an electrical resistor shape is modeled using such an attributed graph. Arcs of the graph represent segments, while the nodes are used to represent vertices. Labels are used to further specify the arcs.

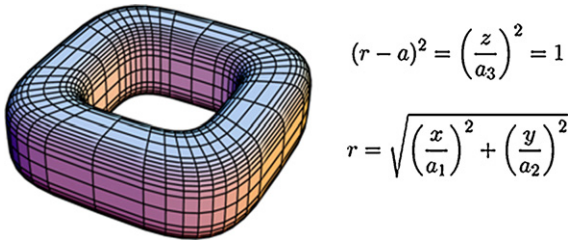
3.1.5. Topology

For design problems whose elements can be instantiated several times, its cardinality can be defined as a topology description. It reflects the number of instances of one element present in the artifact. This variable allows controlling the creation of elements instance when the design problem is one of generating topologic structures. The cardinality of a topology can be constrained by defining the number of element instances in one design artifact. The concept of cardinality also applies to topology relations. In this case, the number of allowable relations is determined by its cardinality. For example, in the case of the gear device, a gear could be connected with more than one gear. If so, the cardinality of this topology relation determines how many gears can be connected one to another.

3.2. Relations

Different types of models are used in design to describe relations. Their characteristics determine the approaches required to handle them, and are here restricted to three basic types: algebraic, differential and logic models. However, this set is not restrictive as others can also be considered.

Topology relation might be used in two different places in a design formulation. The first is to define the set of logic states to hold in the design. The second is as part of the embodiment being designed. In the first, algebraic models using element descriptions are commonly used. Constraint solving techniques have a long history, which makes managing these types of relation feasible with existing methods. For the second, logic models are better suited. In this case, the relation's cardinality has to be taken into account, as the number of relations admissible in the design (number of instances of the relation) is unknown and therefore subject of design. In [11], an ontology of eight basic topology relations is presented through which all possible configuration can be modeled. Also, the bases are established for qualitative reasoning with topological relations using propositional logic models.



$$(r-a)^2 = \left(\frac{z}{a_3}\right)^2 = 1$$

$$r = \sqrt{\left(\frac{x}{a_1}\right)^2 + \left(\frac{y}{a_2}\right)^2}$$

Fig. 6. Super quadric of a Toroid [10].

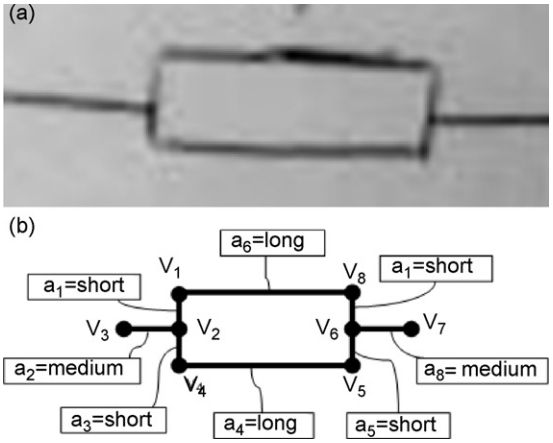


Fig. 7. Example shape graph of electric resistor symbol. (a) Handmade resistor symbol. (b) Graph representation of resistor symbol.

Physical coherence constraints and objective functions are commonly modeled with algebraic relations. In some cases, when the descriptions are symbolic rather than numeric, logic models can be assembled. First order logic and propositional logic models are the most common types of logic models in design.

Analysis relations are often modeled by a combination of all three types. Analytic methods are usually a combination of algebraic and logic models, while simulations make use of numeric methods to solve differential equations.

4. Common design problem formulations

Depending on the type of models involved in the formulation, different design categories – families – can be enumerated. This section describes four common types: parametric, configuration, layout and shaping. In Table 1 these design problems are shown together with the types of models used for its representation.

4.1. Parametric design

When the design problem does not exhibit complex spatial, topologic and shape requirements; and all possible solutions adhere to a common template, it is possible to simplify the problem by modeling the artifact by a set of parameters (see Table 1). In this case, problem solving becomes the process of assigning values to parameters in accordance with the requirements, constraints, and optimization criterion. At present, several algorithms exist for solving this type of problems, as for instance Genetic Algorithms (GAs), Simulated Anealing (SA), Evolutionary Algorithms (AEs), etc.

Table 1
Common design problems.

Information contents		P		C		L		Sh	
Description	Model	E	S	E	S	E	S	E	S
Parameter	–	+	+	+	+		+	+	+
Space	–	+	+	–	+	+	+	+	+
Field	–	–	–	–	+	+	+	–	+
Shape	Geometry	+	+	–	+	+	+	–	–
	Polynomial	–	–	–	–	–	–	+	+
	Constituted Graph	–	–	–	–	–	–	+	+
	Primitives and relations	–	–	–	–	–	–	+	+
Topology	Elements	V		V		F		F	
(Cardinality)	Relations	F		V		V		F	

P = parametric, C = configuration, L = layout, Sh = shaping. E = embodiment, S = scenario, + = present, – = absent, F = fixed, V = variable.

4.2. Configuration design

For design problems that can be modeled in terms of predefined design elements and known topologic relation, the design process consists of assembling and configuring design elements. In this case, shape descriptions and spatial descriptions are not the main subject of design, as shown in Table 1. Solutions need to satisfy design requirements and constraints, and approximates some, typically cost-related, optimization criterion. Configurations can be generated by either instantiating new relation types, or by generating new elements in the topology. Grammatical approaches are very common in the generation of configurations. Startling and Shea [12] developed a parallel grammar for design synthesis of mechanic clocks. An FBS design model of the clock was produced to map the possible Functions to embodiment Structures. A Function grammar (defining the connectivity between Functions) and a Structure grammar (based in the topologic relations of the clock) are used simultaneously to generate solutions.

4.3. Layout design

Determining placement locations for components within a product housing or container is, in short, the goal of layout design. The embodiment elements are described by geometric functions and spatial attributes, while scenario element shape impose constraints, as it can be seen in Table 1. Characteristic to layout design is to count with multiple local optima, space discontinuities, high number of components, constraints, and multiple objectives. All this, makes it a difficult design problem to solve. In [1] references are given to different approaches for solving these types of problem.

4.4. Shaping

Consists of determining the shape of the embodiment elements. Solutions are generated by either defining new mathematic relations or by assembling new graphs structures. Shape grammars have been successfully used in the generation of elements shapes, as reported in [1]. They consist of construction rules that determine how shape primitives can be bounded to produce new shapes. McCormack et al. [13] developed the Buick Grammar, used to generate novel Buick forms. Super quadrics are used in [14] to recognize shape features in the CS of cooling systems for injection molding.

5. Summary

Structure and models for routine design problems were discussed in this paper. Designs are structured in a problem class and a problem instance. A problem class is assembled by defining classes of elements and relations, and interrelating them into one formulation. Elements are differentiated into embodiment (which are subject of designing) and scenario (which model the environment in which the design exist). A problem instance is found by instantiating requirements on both, embodiment and scenario elements. This results in a formulation in terms of initial states, goal statements, constrained states and constraint parameters. Semantic Networks are used to graphically represent design structures and facilitate the development of CS processes for problem classes. Models for representing and generating artifacts are classified into five types: parametric, space, fields, shapes and topology. Each of these are described by mathematical schemes for modeling them.

Results indicate the framework is useful in:

1. Identifying families of routine design problems by analyzing the structures and models used for its formulation.
2. Defining CS processes to automate the generation of design solutions.

Software implementation of this framework is current subject of research. The goal is counting with a computer program where users formulate their design problems by using the structure and models presented in this paper. Then, a reasoning engine would search for adequate CS processes (as in Section 1) by analyzing the structures and models used in the problem formulation. To accomplish this end, data structures should be such that a problem formulation is independent from its solution finding strategies and algorithms. This would guarantee the reusability of elements and relations such that they can be used in new problem formulations.

Acknowledgments

The authors gratefully acknowledge the support of the Dutch Innovation Oriented Research Program 'Integrated Product Creation and Realization (IOP-IPCR)' of the Dutch Ministry of Economic Affairs.

References

[1] Cagan, J., 2001, Formal Engineering Design Synthesis, Cambridge University Press.
[2] Campbell, I.C., Cagan, J., Kotovsky, K., 2003, The A-Design Approach to Managing Automated Design Synthesis, Research in Engineering Design, vol. 14. Springer-Verlag. pp. 12–24.
[3] Cagan, J., Campbell, M.I., Finger, S., Tomiyama, T., 2005, Framework for Computational Design Synthesis: Model and Applications, Journal of Computing and Information Science in Engineering, ASME, 5/3: 171–181.
[4] Simon, H.A., 1973, The Structure of Ill-Structured Problems, Artificial Intelligence, 4/3: 181–201.
[5] Greeno, J., 1978, Natures of Problem-Solving Ability, Handbook of Learning and Cognitive Processes, vol. 5. Lawrence Erlbaum, Hillsdale. pp. 239–270.
[6] Goel, V., Pirolli, P., 1992, The Structure of Design Problem Spaces, Cognitive Science, 16/3: 395–429.
[7] Gero, J.S., 1990, Design Prototypes: A Knowledge Representation Schema for Design, AI Magazine, American Association for Artificial Intelligence, 11/4: 26–36.
[8] Jauregui-Becker, J.M., Tragter, H., Kokkeler, F.G.M., 2007, On the Definition of Parametric Design Problems for Computational Synthesis, in: Proceedings of the ICED07, 237.
[9] Krishnamoorthy, C.S., 1996, Artificial Intelligence and Expert Systems for Engineers, CRC Press, Inc..
[10] Jaklic, A., Leonardis, A., Solina, F., 2000, Segmentation and Recovery of Superquadrics Series, Computational Imaging and Vision, 20:12–39.
[11] Cohn, A.G., Hazarika, S.M., 2001, Qualitative Spatial Representation and Reasoning: An Overview, Fundamenta Informaticae, 46/2: 29.
[12] Starling, A.C., Shea, K., 2002, A Clock Grammar: The Use of a Parallel Grammar in Performance-based Mechanical Design Synthesis, in: Proceedings of the ASME International Design Engineering Technical Conferences, 294.
[13] McCormack, J.P., Cagan, J., Vogel, C.M., 2004, Speaking the Buick Language: Capturing, Understanding and Exploring Brand Identity with Shape Grammars, Design Studies, 25:1–29.
[14] Li, C.L., Li, C.G., Mok, A.C.K., 2005, Automated Layout Design of Plastic Mold Cooling System, Computer Aided Design, 37:645–662.