A Generic Synthesis Algorithm for Well-Defined Parametric Design

W.O. Schotborgh, F.G.M. Kokkeler, H. Tragter, M.J. Bomhoff[1], F.J.A.M van Houten

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

[1]Department of Applied Mathematics, Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, Enschede, The Netherlands

Abstract

This paper aims to improve the way synthesis tools can be built by formalizing: 1) the design artefact, 2) related knowledge and 3) an algorithm to generate solutions. This paper focuses on well-defined parametric engineering design, ranging from machine elements to industrial products. A design artefact is formalized in terms of parameters and topology elements. The knowledge is classified in three types: resolving rules to determine parameter values, constraining rules to restrict parameter values and expansion rules to add elements to the topology. A synthesis algorithm, based on an opportunistic design strategy, is described and tested for three design cases.

Keywords:

Knowledge-based synthesis, Descriptive models, Parametric design

1 INTRODUCTION

Known algorithms to automate the synthesis phase of design include mathematical constraint solvers and inference engines. These algorithms require the knowledge rules and variables in a specific form, such as (in)equalities describing a continuous solution space.

In practice, design parameters are a mix of continuous (e.g. length), discrete (e.g. DIN norms), set-based (e.g. materials) and predicates (e.g. corrosive environment, dynamic load). The (knowledge) rules that model these design objects are often a mix of non-linear, discrete, discontinuous, non-monotonic and (fuzzy) logic and algebraic mathematics. From a computational perspective, developing a generation algorithm for such a model is not easy. Yet, from a human design point of view, the 'design' of a spring is a relatively simple process, often regarded as a search process instead of design. It seems that the complexity of the rules is of little importance to humans.

What is the difference between the human and computational approach? This paper proposes a (generic) model of the design artefact and knowledge rules that enables a human-like approach to solution generation, independent of the computational complexity. Modelling designs and knowledge in a standardized way is beneficial because it provides a common language to compare designs, knowledge and algorithms. It allows development of generic methods and algorithms, e.g. for solution generation and optimization, software architectures and class-libraries. The modelling standard is developed within a larger scope to build synthesis tools quickly for a wide range of industrial engineering design processes.

The idea behind the generation algorithm, which came from cognitive design research, is discussed first. Then, a formal description of a design artefact is given in terms of parameters and elements. Next, the knowledge rules that are used during solution generation are discussed. Having this formal description enables a generic synthesis algorithm to automate the solutions generation process by operating on a knowledge base. This algorithm is suitable for automation, as demonstrated in section 8, where three distinctly different designs are modelled and the synthesis phase automated: a flat belt drive, an optical chamber of an x-ray fluorescence instrument and a baggage handling system.

This paper focuses on well-known parametric designs: the degrees of freedom in terms of parameters and topologies are explicit, as well as the relations between these degrees of freedom.

2 A MENTAL MODEL OF DESIGN KNOWLEDGE

This section discusses a strategy that a human designer uses to find candidate solutions during design, and the role knowledge plays during synthesis. From this, a model for design knowledge and a principle for a solution generation algorithm are derived.

From cognitive design research, "design is most appropriately characterized as a construction of representations. The initial representation is formed by the requirements, and through a series of transformations (e.g. replicate, add, detail, concretize, modify and substitute) develops towards its final form" [1, pp. 131]. The order in which parts of the representations are modified is described by a strategy. One human design strategy is called the "structured decomposition strategy" [1], where the predictable paths from input to output are used during synthesis.

Using this approach for the generation algorithm of compression springs led to an approximately 16-layer deep if-then tree to account for all possible sets of design requirements. This means a considerable amount of software development effort, and the addition of a single design parameter results in significant extra work.

Another strategy is described as opportunistic, where it depends on the current state of the design and available knowledge to decide on that moment what to do. The particular non-systematic character is attributed to the fact that designers, rather than systematically implementing a structured decomposition strategy, take into consideration the data which they have at the time. This focuses on their knowledge, the state of their design in progress, their representation of this design and the information at their disposal [1, pp. 125-126].

2.1 Object-oriented knowledge

We can project the above view of design to the class of well-defined parametric designs. Knowledge rules (e.g. equations) can be modelled as parameter-oriented: aimed at resolving a parameter value. This is the first step of the Role-Limiting Method [2], where the steps to generate design are 1) to extend partial design, 2) to check if constraints are fulfilled and 3) to repair any violations. Knowledge rules that decide if the extension is allowed are modelled similarly: for a specific parameter, the allowed values are determined

If the knowledge rules to extend and constrain is modelled around the design degrees of freedom, an opportunistic strategy can counsel all design parameters to decide which rules to execute, without a premeditated plan. This implements the object-oriented paradigm for knowledge [3, 4]. This allows a clear interface that makes the content of the knowledge (e.g. algebraic, logic, random) independent of the function (e.g. to resolve and constrain).

2.2 Opportunistic design strategy

Opportunistic is described as exploiting immediate opportunities, in an unplanned way. If the goal of design is to determine an allowed value for each parameter, this strategy can be a starting point. Design knowledge is used to refine the implementation. In this paper the concept of ‘certainty’ describes the level of confidence with which a parameter value can be determined by knowledge: calculating a value using an equation has a higher certainty compared to random guessing, for the same parameter. The strategy proposed in this paper determines parameter values following the highest certainty, e.g. first user requirements, then equations and first order logic, estimations, fuzzy logic and finally random value generation. What parameter to resolve next depends on the current state of the embodiment (available information) and available knowledge. A formal notation of a well-defined design in terms of topology elements, parameters and knowledge is discussed in the following sections.

3 WELL-DEFINED PARAMETRIC DESIGN

The class of well-defined parametric design ranges from machine elements to industrial products, such as a transport network for baggage handling systems. This class has predictable parametric and topological degrees of freedom, which can be quantified and fully parameterized. The knowledge rules that govern the degrees of freedom can be made explicit.

3.1 Design process

The design process can be modelled as depicted in Figure 1. The process begins with a set of requirements, stating what the designer wants to design. This set of information is divided into four categories:

1. embodiment requirements: constraints or preferences for the design object/artefact;
2. performance requirements: a quantitative measure of quality, determined by the analysis method(s);
3. scenario description: states in what (worst-case) situation the design is analyzed to determine the performance;
4. **engineering preferences**: a subjective scaling between the performance requirements, to incorporate the design goal with more subtlety.

These requirements are the input for the synthesis phase that generates an embodiment. This is a description of the design artefact that is suitable for analysis. This embodiment enters analysis, together with the scenario. The outcome of the analysis method is the set of performance indicator(s). Evaluation takes these and the requirements into account, deciding what to do next:

1. an embodiment seems promising, an adjustment can be made to it, after which it re-enters analysis;
2. an embodiment does not meet the requirements, nor is it expected to. It is abandoned and synthesis is initiated again;
3. the requirements are met, the embodiment is added to the solution list.

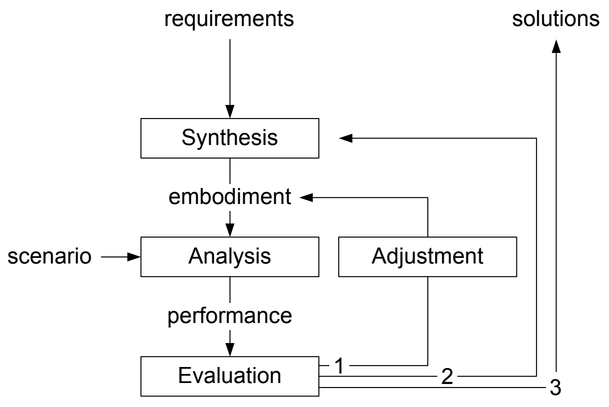

Figure 1: design process

A consistent description of a design process for a specific design includes all relevant information on one or more distinct levels of abstraction. This requires definition of the content of the modules, and their information input/output. An analysis-oriented decomposition is used to acquire this information, discussed next.

3.2 Analysis-oriented decomposition

An analysis-oriented approach is used to decompose a design process and identify the information content of the model of Figure 1.

This decomposition begins with grouping and prioritizing dominant performance indicators within a design process. Because each performance indicator is calculated and quantified by means of an analysis method, these methods themselves are identified. An analysis method is used to identify the parameters of the embodiment, performance and scenario.

The objective of the synthesis phase is to generate an embodiment. In order to model the knowledge and an algorithm for embodiment generation, first a formal model of an embodiment is proposed.

4 EMBODIMENT

An embodiment is a representation of the design object, suitable for analysis. It consists of a hierarchical tree of elements. First, a description is given of a single element, after which the topology of multiple elements is taken into account.

4.1 Parameters and elements

A parameter is an information entity that will receive a value during the synthesis process, i.e. a degree of freedom. Parameters can be of different types, e.g. discrete, continuous, integers and predicates.

Parameters are grouped within elements, so that the element ‘compression spring’ is described by parameters ‘material’, ‘wire thickness’ and ‘length’. During synthesis, multiple instances of the same element are allowed, e.g. two compression springs in a machine. Although these elements have the same parameters, these parameters can have different values. We say, that both elements are of the same 'type' (i.e. compression spring) but have different instances. The description of an element in terms of parameters $p \in P$ is associated to an element type $t \in T$. We have separate types for cogwheels, levers and springs.

Instantiations of an element type are made during synthesis to form the elements $e \in E$. The parameter values of this instance e are represented by the following vector:

$$v_e(p) \qquad p \in P_t$$

For notational convenience, we define $v(p) = \lambda$ to denote that parameter p has not yet been assigned a value. How these parameters receive their values is discussed in section 5.

Note

Embodiment parameters are the minimal set that describes the design artefact with sufficient detail to be analyzed. Beside embodiment parameters, auxiliary parameters are introduced that aid the understanding and design intent, e.g. preferences, ratios.

4.2 Elements and topologies

A topology is defined as a hierarchical tree of elements $e \in E$. During synthesis of a topology, each element e receives a (possibly empty) set of sub-elements $s_e \subset E$.

These sets, of course, have to respect the normal tree semantics, i.e. no element can be its own sub-element, neither directly, nor indirectly. Furthermore, except for a single root element that represents the complete design, we demand that each element has a unique super-element.

A (partial) embodiment is thus represented by the set E of elements in it, together with their associated parameter values v and hierarchical structure s. A partial embodiment is thus represented by a vector (E,v,s).

5 KNOWLEDGE

Apart from the embodiment it generates, a synthesis module is also defined by the knowledge it uses. The knowledge is organized using the object-oriented paradigm, such as discussed by [3]. The objects in question are parameters p and element types t. Each type t contains synthesis knowledge about its own parameters, divided into three types of rules:

- **expand rules** that prescribe the creation of new sub-elements;
- **resolve rules** that govern the assignment of values to parameters;
- **constrain rules** that restrict the possible values of parameters.

The first two of these transform a partial embodiment into another, more refined (partial) embodiment. Constrain rules on the other hand limit the set of allowed values for a parameter.

For an element type $t \in T$, we have R_t, X_t and C_t representing respectively its resolve, expand and constrain rules. It is of importance to note that these rules are the same for every instantiated element e from t, i.e. two compression springs possess the same knowledge.

The expand, resolve and constrain rules share the following properties:

- **object**: the parameter or element type to operate upon;
- **conditional set**: the set of parameters that are required to have a (possibly specific) value, possibly from other elements in the topology;
- **action**: the operation on the object(s), either resolving its value or sub-elements. This is not necessarily a deterministic mathematical algorithm; it could also take the form of a fuzzy logic system or external application.

A more detailed discussion of the rules is given next.

Expand rules

Expand rules `grow' the hierarchy of elements by transforming a partial embodiment (E,v,s) into a new partial embodiment (E',v,s'). This is done by determining the set of sub-elements s'_e for an element $e \in E$ for which $s_e = \lambda$ and adding these elements to E to form E'. A single expand rule can add multiple types of elements, or multiple instances of the same element type.

Resolve rules

A resolve rule transforms a partial embodiment (E,v,c) into a new partial embodiment (E,v',c) that has one less free parameter. I.e., to transform v to v', the value of a parameter p for some element $e \in E$ is fixed from its state λ in v. This implies resolve rules can only be applied to fix values of parameters that had not already been fixed before.

A random generator is a resolve rule that can be applied to many types of parameters: in case of floating point or integer parameters, it will generate a value within the solution space. In case of e.g. a material, it can randomly select an allowed material.

Constrain rules

Constrain rules govern the boundaries of what is possible within a design problem. It returns a set of allowed values for a parameter: c_p. At any time during the synthesis process, the value of any assigned parameter must be a member of all of the sets of allowed values produced by applicable constrain rules.

5.1 User requirements

User requirements are (combinations of) additional rules without a conditional set, i.e. always and immediately executable. This is the starting point for a synthesis process, and any embodiment must satisfy these rules.

Examples are: length = 10.0, thickness ≤ 5.0mm, M4 ≤ bolt size ≤ M20, material ≠ {copper, aluminium, gold}.

5.2 A solution

A solution is an embodiment that satisfies the following conditions:

$$s_e \subset E \qquad \forall e \in E \quad (1)$$

$$v_e(p) \neq \lambda \qquad \forall e \in E, p \in P_t \quad (2)$$

$$v_e(p) \subset c_p \qquad \forall e \in E, p \in P_t \quad (3)$$

I.e. (1) topology fully expanded, (2) no parameter value is unresolved for any element and (3) each parameter value lie within the allowed set, for each element.

6 SYNTHESIS ALGORITHM

A synthesis algorithm is discussed that consists of a specific part and a generic part. The specific part contains the description of element types (parameters and knowledge). The generic part of the algorithm resolves the parameters using an opportunistic strategy, based on a design mechanism used by a human designer. This step-wise approach is used to expand the topology and resolve parameters, where each step considers the available knowledge and embodiment representation to decide what to do.

An algorithm is presented to generate one solution, given an element description.

Step 0: initialization

The algorithm initialization requires a set of element types T . A single instance is denoted the root element, from which the algorithm starts. User requirements of an element e are superimposed on the knowledge base when they are instantiated.

Loop step 1: constrain check

Goal: test if all parameter are valid.

Execute constrain rules C_t if the conditional set allows it. Test for all parameter:

1. solution space not empty, i.e. $c_p \neq \varnothing$;
2. their value (if resolved) lies within the allowed set, i.e. $v_e(p) \in c_p$.

If all tests are passed: current embodiment representation is allowed. If one test is negative: current embodiment is not allowed. A previous (allowed) representation (E,v,s) is retrieved to proceed.

Loop step 2: complete check

Goal: check for complete embodiment

Test for all elements $e \in E$:

1. $s_e \subset E$;
2. $\mathrm{v_e(p)} \neq \lambda \qquad \forall \mathrm{e} \in \mathrm{E}, \mathrm{p} \in \mathrm{P_t}$

I.e. (1) topology fully expanded, and (2) no parameter value is unresolved for any element. If tests are passed, the embodiment (E,v,s) is a complete representation and the synthesis phase is terminated.

Loop step 3: advance partial embodiment

Goal: execute one expand or resolve rule.

This is done in two phases:

1. explore possibilities;
2. execute rule

The first step tests the conditional sets for the expand and resolve rules of each element. The decision, which one to execute, can be made with or without a strategy. In the most basic form, this is a random selector.

The second step executes the action of the rule, effectively creating a more refined embodiment. An expand rule results in an embodiment (E',v,s') and resolve rule produces (E,v',s). After this action, the loop continues with step 1.

Notes

If backtracking is implemented to solve constrain rule violations, the algorithm effectively implements a non-deterministic version of depth-first search.

7 KNOWLEDGE EXTRACTION

In the development process of any synthesis tool, an important phase is that of knowledge extraction. The functionality of the tool is determined here, as well as the knowledge that will be used to generate solutions. The object-oriented knowledge organization paradigm enables efficient extraction and implementation of design knowledge from expert designers or literature sources.

The goal of knowledge extraction from human or literature source is to model the design process as a consistent and coherent whole with predictive behaviour. This translates to engineering design knowledge modelling as determining the right parameters and knowledge rules. It should enable a third party (algorithm or novice designer) to generate designs.

An analysis-oriented decomposition method first isolates performance indicators of equal importance and their analysis methods. Second, the expressiveness of the analysis methods leads to the identification of the scenario and embodiment parameters. The embodiment description is divided into element types and parameters. The synthesis knowledge determines the formation of elements. Groups of parameters that occur in the same rules are likely to belong to the same element, however his is somewhat subjective. Also, the physical model can serve as an indicator.

The knowledge rules for synthesis are extracted, or formalized, by studying for each parameter:

1. how a parameter value is determined (= resolve rule);
2. what constrains are relevant (e.g. geometric, manufacturability, experience).

When dealing with topologies:

3. how to determine the sub-elements (= expand rule).

And possibly, to guide the parameter selection process:

4. In what order the parameters or elements are resolved (= a strategy).

If an (expert) designer is available, an interview style knowledge extraction can be done. This knowledge extraction process is eased due to the explicitness of the parameters.

A method to determine a coherent model is an analysis-oriented approach for a new (and unfamiliar) design process is discussed in more detail in [5].

8 EXAMPLE

A number of cases have been used to verify the approach described in this paper. Three examples are illustrated here. The design processes of these cases were modelled using the approach described in this paper. Implementation of the described synthesis algorithm resulted in automatic generation of embodiments, independent of the user requirements. Some characteristics of the knowledge used during synthesis are depicted in table 1. This table indicates the relative complexity of the three cases, as far as synthesis knowledge is concerned.

A complete synthesis tool for design of flat belt drives (Figure 2) is developed. Analysis methods and synthesis knowledge is extracted from a mechanical engineering handbook. Embodiment parameters are belt material, width and thickness, disc diameters and axis distance. Auxiliary parameters are transmission ratio, overall length, enclosed arc around smallest disc, 2 'utility factors' and length of the belt. Scenario description is given in terms of rotation speed, torque and power, for both discs.

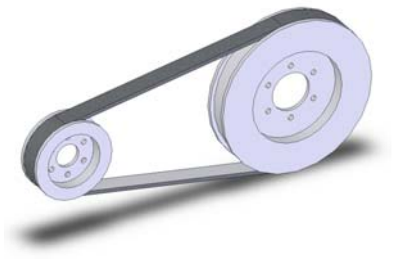

Figure 2: flat belt drive

A second example is the optical chamber of an x-ray fluorescence instrument, Figure 3. This chamber is the heart of an instrument that determines the chemical composition of a material. The x-ray source radiates the sample material through a diaphragm. The sample, contained in a holder, expels characteristic photons into a detector. This in turn reveals the composition of the sample.

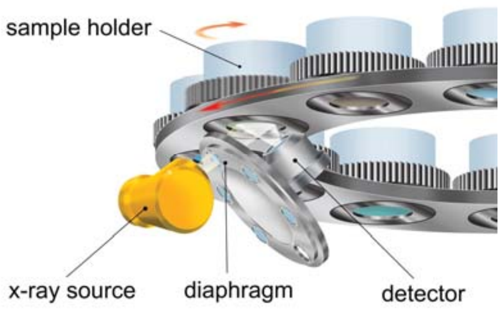

Figure 3: optical chamber [copyright PANalytical BV]

The synthesis phase of the design process is automated for baggage transport networks, Figure 4. This design consists of a modular arrangement of transport units and equipment that process the pieces of baggage. The knowledge base of this design is continuously being expanded, so table 1 contains a snapshot. Embodiments for this network contain many hundreds of transport connections.

Figure 4: transport network
[copyright VanderLande Industries]

CONCLUSION

The presented approach enables modelling of a design process, design artefact and synthesis knowledge for well-defined parametric design. The validity ranges from machine elements and product components to transport networks. The modelling process (extraction and structuring) is aided by the explicitness of the information content. The model is suitable for automation and development of computational synthesis tools as well as documentation of design knowledge.

ACKNOWLEDGEMENT

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] Visser, W., Designing as Construction of Representations: A Dynamic Viewpoint in Cognitive Design Research. Human-Computer Interaction, 2006, Volume 21, pp. 103-152, Lawrence Erlbaum Associates, Inc.

[2] Studer, S, Benjamins, V.R, Knowledge Engineering: Principles and methods, Data & Knowledge Engineering 25, pp. 161-197, 1998, Elsevier Science

[3] Bento, J., Feijó, B., Smith, D.L., Engineering design knowledge representation based on logic and objects. Computers & structures, 1997, vol. 63, no. 5, pp. 1015-1032, Elsevier Science Ltd, Great Britain

[4] Zhang, W.Y., Tor, S.B., Britton, G.A. prototype Knowledge-Based System for Conceptual Synthesis of the Design Process. Int J Adv Manuf Technol, 2001, 17: 549-557, Springer-Verlag London

[5] Schotborgh, W.O., Tragter, H., Kokkeler, F.G.M., van Houten, F.J.A.M., A Method to Translate an Engineering Design Process into a Structure for Computational Synthesis, 16th International Conference on Engineering Design, Proceedings of ICED'07, 2007, pp. 65-66, Paris

Table 1: characteristics of the knowledge base

	Belt drive	Optical chamber	Baggage handling system
parameters	18	46	191
element types	1	8	48
expand rules	0	4	45
resolve rules	28	22	80
constrain rules	21	20	18