INTERNATIONAL DESIGN CONFERENCE - DESIGN 2008
Dubrovnik - Croatia, May 19 - 22, 2008.

DESIGN 2008

TOWARDS A DEVELOPMENT METHOD FOR COMPUTATIONAL SYNTHESIS SYSTEMS

W.O. Schotborgh, F.G.M. Kokkeler, H. Tragter, M.H.L. Röring and F.J.A.M. van Houten

Keywords: computational synthesis, knowledge, engineering

1. Introduction

A computational synthesis system (CSS) is a software system that automates the generation phase of a design process. It generates solutions based on knowledge rules and design requirements, presenting an (interactive) overview of the solution space. This enables an engineer to find the best design through user-friendly exploration of design requirements and candidate solutions. Research yielded advanced possibilities for synthesis techniques including self-learning multi-agent systems with multi-objective optimization [Cagan 2005], on conceptual and embodiment level [Starling 2004]. During the overall design process, such tools should increase insight in the design constraints and guide the designer from requirements to feasible products [Ullman 2002]. Previous research has shown that one generic CSS for all of engineering is out of reach, for now. The required expressiveness and versatility of such a tool [Braha 2003] and fundamental understanding of the synthesis process [Tomiyama 2007] remain challenges to be solved.
A CSS is rarely found in everyday life of engineers, even for the most basic engineering tasks such as quantitative routine design of machine elements. Why is this? A possible explanation is that the development process of such a tool is an unpredictable investment. As noted by Cagan et al. [Cagan 2005], the development of a CSS for a new design case "has not received much attention in the literature, since most computational synthesis methods are developed to solve a particular design problem". Ideally, a CSS development process should be fast, efficient and supported by as much generic software as possible to minimize software development effort. This paper proposes a definition of generative knowledge to allow documentation of a CSS development process. This process begins with knowledge acquisition and representation. The representation includes the design artefact and sufficient knowledge to generate candidate solutions. The scope and approach are discussed in more detail in the following sections.

2. Scope

This paper focuses on quantitative routine design, which includes machine elements (e.g. bearings, springs), product components and layouts. This class has predictable degrees of freedom on parametric and topology level. The design knowledge is available, either explicitly documented or tacit through expert designers.
In everyday life of engineers in industry, the class of quantitative routine design represents a significant part of their work. Without presenting proof, the authors assume that automating the generative phase for this class of design is an important step in making CSS a commodity in industry. The scope of this paper is the generative phase in quantitative routine design.

3. Approach

The approach is to divide the candidate generation module of the CSS into two parts: a generic part that is equal for all CSS, and a specific part that changes per design case. This approach is not new in KBE: Expert Systems have a division between the inference engine and knowledge base. Mathematical constraint solvers have a division between solver and e.g. systems of equations. This means the design problem has to be modelled in a specific format before it can be processed by the solver. This reduces the knowledge modelling freedom and introduces the challenge of translating design knowledge into the specified format, see section 3.1 for an example.
The approach in this paper is to use a solver that does not operate directly on the content of the rules. Instead, an interface is defined using three basic operations during the generative process. The solver controls the basic operations, and the knowledge rules execute these operations. The solver only receives qualitative information about the content. This allows a high level of knowledge modelling freedom within the content of the rule: equalities, inequalities, reasoning, random generation, etc. It can also include external applications such as database search, FE analysis or other mathematical constraint solvers (or a complete CSS).
First, a definition of design representation is proposed in section 4. Based on this definition, section 5 introduces three generic basic operations during the generation process of candidate solutions. The generative algorithm that controls these operations is discussed in section 6. A knowledge acquisition method is proposed that extracts the minimum set of knowledge that is required for candidate generation: section 7. Section 8 validates the approach for three design cases and section 9 concludes the paper.

3.1 Example

If a design process is fully defined in terms of algebraic mathematical relations, algorithms exist to generate solutions. Examples include symbolic manipulation, numerical root finding and optimization techniques: the world of mathematics. However, even for a simple engineering problem such as the design of a compression spring, this is not trivial. (1) is a generative knowledge rule for the estimation of the wire thickness ‘d’ for compression spring.

$$d \approx k_1 \cdot \sqrt[3]{F \cdot D_e} \tag{1}$$

This rule is used to resolve the parameter value for ‘d’ if the required force ‘F’ and outside diameter ‘D_e’ are known, with k_1 a discrete factor from the set {0.8, 1.2, 1.4}, depending on the magnitude of F. Although it is certainly possible to model this mathematically, it is not straightforward.
However, from a design perspective, this is quite simple: if ‘F’ and ‘D_e’ are known, then ‘d’ can be estimated. The basic operation of this rule is to resolve parameter ‘d’. The complexity of the relation itself is of little importance.

4. Design artefact representation

The design artefact representation describes the expressiveness of the CSS. The design artefact is described as a set of parameters and topological elements. A parameter is an information entity that receives a value during the generation process, i.e. a degree of freedom (e.g. discrete, continuous or integer values, predicates). Parameters are grouped within elements to allow the creation of topological structures of artefacts. Within a candidate representation, multiple instances of the same element are allowed. Although these elements have the same parameters, the parameters *values* can be different. We say, that both elements are of the same ‘type’ (i.e. two wheels on a bike) but have different instances (front wheel and back wheel) with different parameter values (front wheel radius = 300mm, back wheel radius = 350mm).

5. Generative rules

During generation of candidate solutions, each parameter from the design artefact has knowledge to determine a value and check the validity of this value. Following the object-oriented knowledge paradigm [Bento 1997], this generative knowledge can be explicitly documented in a parameter-oriented way. This can be done with three basic operations during candidate generation, for parameters and elements:

1. resolve: resolve a parameter value (e.g. equalities);
2. constrain: constrain the allowed parameter values (e.g. inequalities);
3. expand: add element(s) to the topology.

A resolve rule reduces the problem space, while an expand rule introduces new degrees of freedom that require resolving, thus expanding the problem space. Constrain rules are used to check if the current set of parameters is valid. The basic operations are independent of the content within the rule, allowing a knowledge independent algorithm of the generative process, discussed in section 6. A more detailed discussion on this formalization is found in [Schotborgh 2008].
The content of the rules is solver neutral, with the restriction that is an explicit and computable function description.

6. Generative algorithm

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" [Visser 2006]. The order in which parts of the representations are modified is described by a strategy. One possible strategy is described as opportunistic: it depends on the current state of the design and the available knowledge, to decide what to do next. 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 [Visser 2006].
For such an opportunistic design strategy an algorithm only takes into account wheter a rule can be executed, not how it is executed. This indicates a difference between the content and operation of a knowledge rule.
Opportunistic design strategies can be implemented into algorithms such as the Role-Limiting Method (RLM) [Studer 1998]. The RLM algorithm consists of three steps: expand representation, check constraints and repair any violations. The generative algorithm proposed in this paper, is similar to the RLM, but more specific to our definition of generative knowledge.
The algorithm operates on a knowledge base, requesting information on the executability of the resolve, expand and constrain rules. All possible generative operations are gathered, after which one rule is selected from execution. A pure opportunistic strategy chooses an option randomly, but more sophisticated strategies can be implemented. After a modification is made, all allowed constrain rules are executed to check if the modification introduced any conflicts. In case of a conflict, this should be repaired, e.g. by retrieving a previous representation from stored history. After this, or in case of no conflict, another check is performed to see if the representation is complete. If this is not yet the case, the algorithm continues.
Higher-order knowledge such as experience-based design strategies and heuristics can be included to manage the overall generation process, making the design process less opportunistic and more guided. A more detailed discussion on the algorithm is found in [Schotborgh 2008].

7. Knowledge acquisition process

The knowledge acquisition process to retrieve the artefact description and generative rules is similar for human and literature source. This process begins with entering a new company to develop a CSS for (part of) a design process, and contains the following steps to automate the generative phase:

1. Overview: make a first scan of the company, its products and design process(es) to familiarize with the context, individuals and nomenclature.

2. Selection: select (part of) a design process for further modelling.
3. Level of abstraction: determine an isolated level of abstraction by identifying the performance indicators of equal importance.
4. Analysis formalization: formalize the analysis method, and extract the embodiment parameters. Analysis is the step after candidate generation, to assess the quality of this candidate. Studying the analysis method reveals the minimum description of the design artefact.
5. Generative knowledge: formalize for each parameter the resolve and constrain rules. The topological expansions are described in expand rules.

Combining the candidate models from multiple analysis methods leads to the complete model of the representation on a single level of abstraction. This model can be used to document the knowledge and rational, or to automate the design process on that level of abstraction.

8. Demonstration

Three quantitative routine design cases are used to demonstrate the approach described in this paper. The generative knowledge is modelled and a generic synthesis algorithm is used to generate candidate solutions. Comparison of the generative knowledge is done in section 8.4.

8.1 Machine element

A complete synthesis tool for design of flat belt drives (Figure 1) is developed. Generative 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.

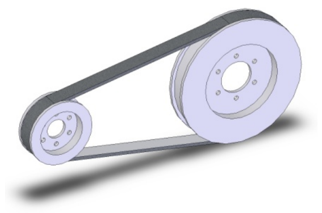

Figure 1. Flat belt drive

8.2 Product component

An example of a product component is the optical chamber of an x-ray fluorescence instrument, Figure 2. 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.

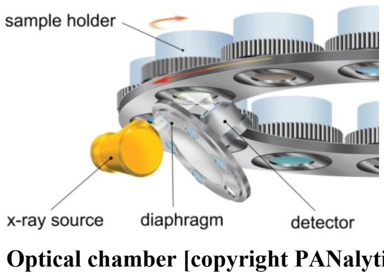

Figure 2. Optical chamber [copyright PANalytical BV]

Besides generative knowledge, application specific decisions rules are also included. This design process is not explicitly documented but only tacitly present in the mind of an expert designer. The knowledge acquisition technique focuses on the content of each generative rule, which delivered sufficient information to enable automatic candidate generation.

8.3 Product layout

An example of product layout with a dynamic topology is the design of baggage transport networks, Figure 3. This design consists of a modular arrangement of transport units and equipment that process the pieces of baggage. Embodiments for this network contain many hundreds of transport connections and generation process is done in several thousand design steps.

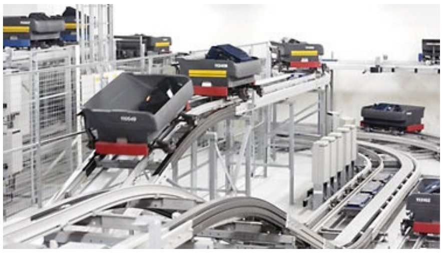

Figure 3. Transport network [copyright VanderLande Industries]

8.4 Comparison

The three design cases are different in terms of number of parameters, elements and generative rules. Table 1 indicates the relative complexity of the three cases, as far as generative knowledge is concerned.

Table 1. Generative knowledge

	belt drive	optical chamber	baggage handling system
parameters	18	46	191
elements	1	8	48
expand rules	0	4	45
resolve rules	28	22	80
constrain rules	21	20	18

9. Implications for engineers

Having a CSS available for a design task enables the user to scan possibilities without having to acquire the knowledge and perform the iterative search. The goal of these systems is not to deliver a fully optimized solution, but rather make a broad scan of the solution space. Instead of trying to make the software more aware of its context, the functionality is made transparent and clear, leaving the selection of the best candidate to the user.

CSS can be directly connected to CAD software to provide input for the CSS and improve communication and discussion with colleagues *after* candidate generation, with a clear view on solutions and alternatives. As an assembly drawing is modified, a new candidate solution can be generated within it, with the same functional specification. If the quality of a design cannot be sufficiently quantified by computers, but has to be judged by visual or tactile analysis, virtual reality and haptic devices can be included in the loop. Although this might slow down the automated process, the increased insight in the solution space aids the optimization process and the quality of the candidates. Experts from different disciplines can be invited into virtual reality to assess and discuss the quality of solution candidates.

With the search process through the solution space being automated, industry can focus more resources on knowledge generation itself to increase the effectiveness of the search. Research activities can be formulated in terms of parameters or elements that require better understanding. During the automatic search for candidate solutions, diagnostic systems can monitor the efficiency of certain rules or parameter estimations, giving clues about the quality of knowledge itself. Machine learning techniques can be used to mine for new knowledge rules, generating new knowledge automatically. The tool guarantees consistent use of knowledge, making this available to novice designers throughout the company, and outside the company for marketing purposes.
A difference with existing FEA applications is depicted in Figure 4. Day-to-day usage of (basic) FEA software is aimed at modelling part of a product to a mesh. This (human) process requires a certain understanding of FEM modelling. Once a mesh is obtained, the software calculates the results, e.g. a displacement field. Validity of this result depends on the quality of the mesh and e.g. boundary conditions. Synthesis software has a different usage: once a design process is abstracted into a knowledge base, the intended usage lies in the *computational* part. Consistency is easier guaranteed since there is no human interaction. The risk of garbage-in, garbage out does not lay with day-to-day usage.

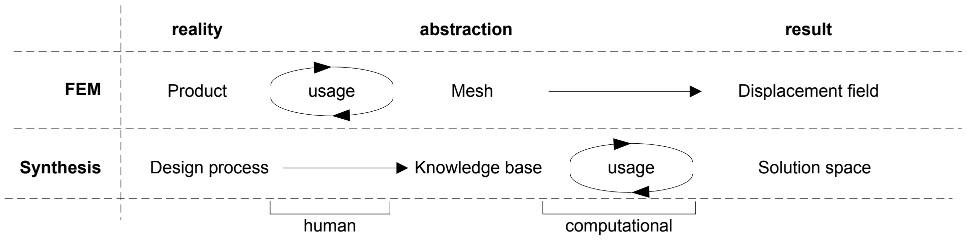

Figure 4. FEM and synthesis

10. Conclusion

A development process for CSS is outlined from knowledge acquisition to automation. The expressiveness of the proposed definition is demonstrated by automating the candidate generation phase of three design cases. A definition of generative knowledge enables development of generic software architectures and algorithms, to reduce the software development effort for CSS.
Having a dedicated CSS enables engineers to design with higher level functionality, where solutions are generated on command. This eliminates the necessity to acquire the knowledge itself and perform iterative search towards a solution. Providing a CSS for a design task allows other engineers to apply that knowledge without having to become experts themselves. This will ease multi-disciplinary design and system-level thinking.

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

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

Braha, D., Reich, Y., "Topological structures for modeling engineering design processes", Res Eng Design, 14(4): 185-199, 2003, Springer-Verlag Londen Ltd.

Cagan J., Campbell, M.I., Finger, S., Tomiyama, T., "Framework for Computational Design Synthesis: Model and Applications", Journal of Computing and Information Science in Engineering, 2005, ASME.

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, 2007b, pp. 65-66, Paris

Schotborgh, W.O., Tragter, H., Kokkeler, F.G.M., M., Bomhoff, van Houten, F.J.A.M., "A Generic Synthesis Algorithm for Well-Defined Parametric Design", proceedings of the 18th CIRP Design Conference 2008, -to be published-, 2008

Starling, A.C., "Performance-based computational synthesis of parametric mechanical systems", PhD thesis, 2004, St John's College, University of Cambridge.

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

Tomiyama, T., Schotborgh, W.O., "Yet Another Model of Design Synthesis", 16th International Conference on Engineering Design, Proceedings of ICED'07, 2007, pp. 83-84, Paris

Ullman, D.G., "Toward the ideal mechanical engineering design support system", Res Eng Design, 13(2): 55-64, 2002, Springer-Verlag.

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

Wouter O. Schotborgh
University of Twente, Faculty of Engineering Technology
Department of Design, Production and Management
P.O. Box 217, 7500 AE Enschede, the Netherlands
Tel.: +31 53 489 2266
Fax.: +31 53 489 3631
Email: w.o.schotborgh@utwente.nl
URL: http://www.opm.ctw.utwente.nl/research/index.html