INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED'09
24 - 27 AUGUST 2009, STANFORD UNIVERSITY, STANFORD, CA, USA

WHY IS DESIGN AUTOMATION SOFTWARE NOT EVERYWHERE?

Wouter Schotborgh, Frans Kokkeler, Hans Tragter, Fred van Houten
University of Twente

ABSTRACT
This paper discusses the question how to increase the amount of design automation software in industry. First, the contemporary industrial context is outlined to motivate the need for automation of parametric routine design problems. Second, the knowledge engineering (KE) activity is identified as a bottleneck that prevents large scale deployment of design automation software. Knowledge engineering is seen as the process to construct a mathematical model of a design problem. Literature and our own research results are used to illustrate that the KE activity for new problem automation is often unpredictable in terms of the activities to execute and time it will take. A correlation seems to be missing between the problem size and the required knowledge engineering effort. Possibly, this is a result from the diversity of design problems and the tacit nature of expert knowledge. A route toward more prescribed KE is suggested by tightly integrating theories from cognitive design, knowledge engineering and constraint satisfaction. The goal is to closely relate knowledge acquisition (interviewing of experts) with the knowledge modeling activity and automation algorithm.

Keywords: design automation, knowledge engineering, method

1 INTRODUCTION
In everyday life, consumers that want to buy a consumer product are supplied with online shops and comparison sites that support the decision making process. Software tools make shopping faster, easier and provide more overview of alternatives.
For engineers, a product creation process often involves a number of routine decision making processes that still consume considerable time. The available aids for routine parametric problems are handbooks, formulas, catalogues and/or Excel-sheets with experience knowledge. Finding the right design takes time and as a result, only a handful of alternatives are considered. The commercially available support tools for decision making for engineering shows a discrepancy with the support for consumers, even for the parametric routine type of problems.
This paper aims to identify the bottleneck that prevents large scale deployment of decision making tools for engineering, and suggest an approach to relieve the bottleneck. The scope is routine engineering design with parametric information and quantitative information. This covers continuous and discontinuous parameters, as well as a mix of linear and non-linear equations, logic and fuzzy estimations, for static and dynamic topologies. The scope includes the design of machine elements, product components and product systems.
The goal of this paper is to motivate the need for a prescriptive knowledge engineering method, and is part of the Smart Synthesis Tools project [1], [2]. The prototypes that are used to illustrate the KE process as bottleneck are explicitly documented design problems from handbooks. Design problems in industry are quite different in the sense that industrial problems have more tacit knowledge and the models are not explicitly known. The handbook-prototypes are used because there still is a KE activity to translate the pages of text into a model that can be automated by an algorithm.

1.1 Paper layout
Software that automates design processes and generates multiple design candidates is well-known. Section 2 briefly discusses the contemporary industrial context to motivate that design automation software is very suitable to be deployed on a large scale.
Section 3 reviews the commercially available support for engineering design and identifies that only little software with automatic candidate generation is found. Still, automating routine design tasks will

save time, improve quality, increase insight and allows designers to quickly scan possibilities. The algorithmic state-of-art is beyond what is needed to automate routine design for parametric problems.
Section 4 illustrates an emerging bottleneck: the process of constructing a mathematical model of the problem, based on the knowledge of expert designers. This process requires information from the knowledge source to be translated into a format that can be automated. Challenges are the interviewing of experts, or analyzing textbooks, and deciding upon the system boundaries, quantity and content of the knowledge rules and ensuring a maintainable knowledge base.
Section 5 discusses in more detail the field of developing knowledge models: the domain of knowledge engineering.
Section 6 suggests an approach to relieve the bottleneck, by integrating the views of the knowledge with models and algorithms of automation. This requires integration of theories from cognitive design, constraint satisfaction and knowledge engineering. The goal is to form a method that offers a continuous development procedure from knowledge source to software system. A prescriptive aid should be provided to prevent the necessity for a knowledge engineer to first possess all required knowledge and then translate this into a mathematical model plus implementation.

2 INDUSTRIAL CONTEXT

We encounter in the world around us an enormous stream of products with constantly changing features and appearances. It seems that a product is (re)designed for nearly every taste, price range and user group. The trend of increasing product diversity has a profound impact on companies, teams and individuals that develop these products [3].
The force that drives the growing supply of consumer products can be traced back to the developing countries. These countries rapidly improve their socioeconomic standard, bringing with it a vast consumer base of hundreds of millions of people. These people gain access to modern products and services which make these markets a tremendous potential for firms to increase revenue.
Many companies change their models of commerce, marketing strategies and product palette in order to take full advantage of the emerging markets [3]. The new type of consumer demands products with low prices but modern performance, tailored to their local taste. One of the answers of multinational companies is to use existing technologies to develop high quality products in great variety [4]. The speed of new product development becomes increasingly important.
The fast development of consumer markets increases the pressure on the engineering design process to reduce time to market [5], [6], [7]. Uncertainties and risks are reduced where possible [8] and information and communication technology is adopted to enhance flexibility, speed and efficiency of the process [9], [10].

3 SOFTWARE SUPPORT FOR ENGINEERING DESIGN

A review of the commercially available software reveals that the majority focuses on the design process steps of analysis, drawing and refining details of established design concepts. Only little software support addresses the creation process of designs. The majority of software requires a fully defined design as input, which forces the engineer to plan ahead and make choices. After a weak point is identified by analysis, this can be corrected in a number of ways. Only little methodology is provided for guidance about what to do next [11], [12] and [13].
Ullman [11] describes an ideal support system for the creation of new products: insight is provided in the relationship between the customer wishes and the available product options. The search for the best design is an automated process, guided by the preferences of the engineer. Support systems present alternative solutions that give an overview of what is possible. This allows experts to use their experience and “design intent” to select the solution that is better than all others.

3.1 Generating designs

Different software functionalities have been developed to support the engineering design. This paper focuses on software that generates multiple design solutions for a set of requirements, a type of software of which a large body of research results are available. Designers specify the required behavior and the software generates possible designs that meet the requirements.
Computational Synthesis (CS) is a field of research that generates and optimizes designs with topological variety.

A research overview of CS is given by Antonsson and Cagan [14]. CS offers support for design processes ranging from shape driven (architectural) design to electro-mechanical systems. The concept of grammars offers a formalization of design synthesis knowledge. The result is a form of production rules, or graph based pattern recognitions, that expand an initial graph into an eventual design. It supports geometric representations, reasoning and emergent shape properties [15]. Generating solutions on both the topological and parametric level can be done using so-called parallel grammars, e.g. for gear design [16].
An example of constraint-based approaches is described for the field of 3D mechanism and linkage system design in [17]. The model of the problem is stated as equalities and inequalities, and multiple sets of values that represent (possible) solutions are presented to the user. Seeing and judging multiple design candidates increase the understanding of the design problem [17]. A constraint-based model describes the problem and direct search optimization is implemented to generate solutions.

3.2 Parametric design

An example of software that generates multiple designs for parametric problems is given in Figure 1. The figure shows the theoretical model and the software tool developed. The models and relations from commonly engineering handbooks are implemented. The tool shows the behavior information that is taken into account and allows the user to specify known parameters, such as the mass of the object or input motion.

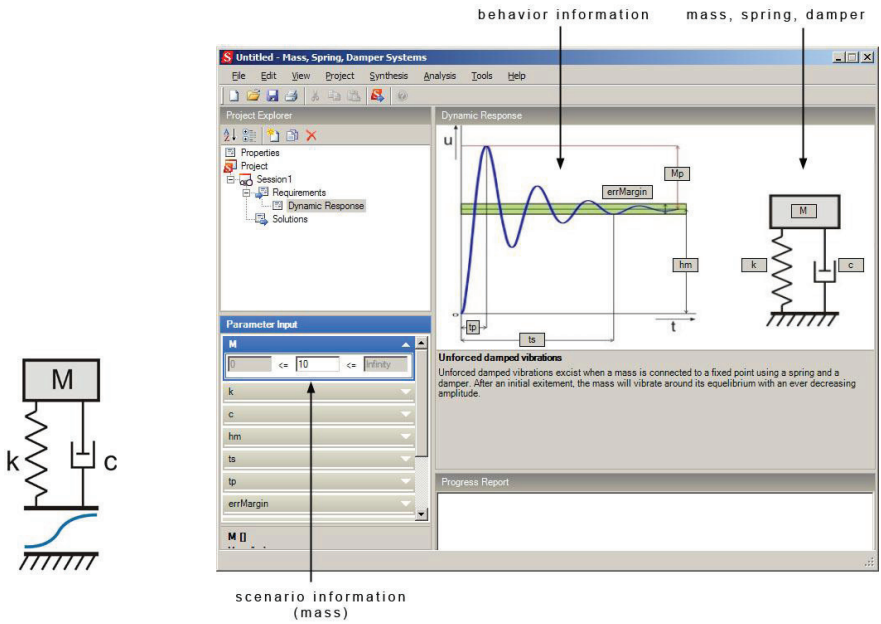

Figure 1: parametric engineering design

The software is used to design a car suspension system. The design goal is to comfortably absorb a bump in the road: the car should go up and down only slightly and quickly stop bouncing. Consider the situation of a car driving up a curb of say 10cm, at a speed of 5km/h.
The designer has to find the right specifications for the suspension system, consisting of a spring and damper in parallel arrangement. The software generates and analyzes multiple designs, visually presenting the solutions in a solution space. The solution spaces provide quantitative information about the possible behaviors of designs. The behaviors of interest are in this case the height and the duration of the bouncing motion. The software allows the designer to see what is possible, and select the solution(s) he/she prefers.

Figure 2, left hand side, depicts the quantitative behavior specifications of about 200 designs: the overshoot on the y-axis and the settling time on the x-axis. Each dot represents a quantified design. The favorable designs are located in the bottom-left corner (the red group). Figure 2, right hand side, shows a plot of the two main design variables: damping coefficient and spring coefficient. In this plot, the same group of best designs is located at the upper section of the solution cloud (the red group). Observing these two plots, the designer sees what springs and dampers will result in a comfortable ride.

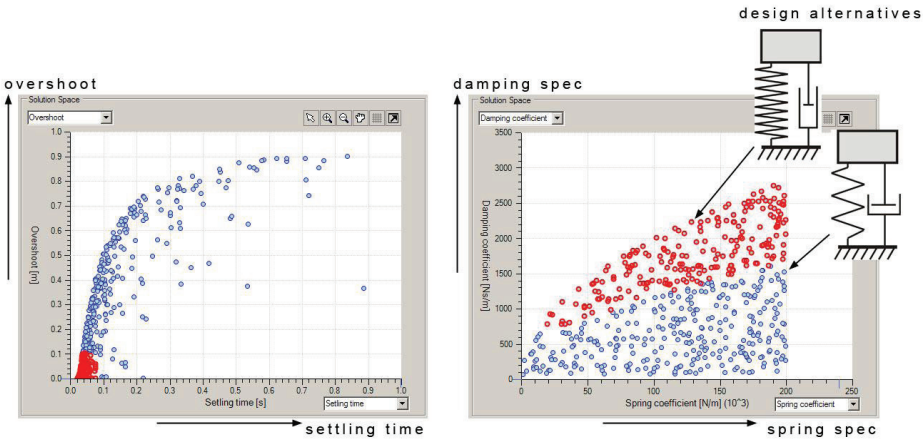

Figure 2: design automation, solution spaces

The insight on the design possibilities is difficult to obtain from theoretical analysis and trial-and-error iterations by hand. The software quickly provides an overview of possibilities, which saves time and steers toward an optimal design because the designer is aware of the alternatives and picks the design that suits him best.

For the example in Figures 1 and 2, the technology has been available for decades. However, such software will not be mass-produced until the development cost lay under a certain limit. For the spring-damper example, the mathematical models of the problem should be acquired from the source in hours. For industrial problems with human experts, the knowledge acquisition and modeling activity should be done also in an efficient, methodological manner.

A methodological approach to acquire models for a diverse range of engineering design has to deal with several different types of knowledge, elaborated further in the following section.

3.3 Design knowledge

The scope of parametric routine design ranges from theoretical problems, machine components to products. The modeling entities and constraints to satisfy are encountered in a variety of ways. Generating designs is generally an under constrained problem: there are more unknown parameters than there are equations. Actually, having equations is already a luxury. Studying design cases from engineering handbooks and industrial settings shows a mix of linear and non-linear algebraic equations, logic reasoning and random estimations with exceptions or uncertainties. These relations can come from scientific formulas or fuzzy experience knowledge. Choices made earlier can influence what rules to apply later on. Designs with topological freedom can include or exclude entire groups of variables and relations.

The parameters that describe a design are a mix of continuous and discontinuous, Booleans and set-based parameters. For example, the parameter “material” is composed of a Young's module, density and melting temperature. One cannot use an equation to determine a value for the Young's module: such a material is not guaranteed to exist.

Another challenge with automatic generation of designs is the unpredictable input information. The user might, or might not, prescribes the value of certain parameters. And this can change each time the software is used. The algorithm could encounter systems of equations, or worse: a system of relations

with all the mixed types described earlier. And in advance one does not know if a solution exists or not. In any case, the algorithm that generates designs has to complete the rest of the design automatically and find solutions, if they exist.
Developing a software module that processes the user input and generates designs accordingly can be done in an ad hoc manner: study the design case and decide how to implement the model and algorithm. The following section uses six prototypes to discuss the ad hoc approach.

4 DEVELOPMENT PROCESS

Illustrating the development of design automation software is done with several prototypes that were developed before 2005 [18] (several years before the tool of Figure 2). Please note, the scientific rigor of said prototypes is insufficient to draw any firm conclusions. I aim to demonstrate the unpredictable amount of effort that is required to model and automate apparently similar design problems.
Comparing the problem complexity and the development effort is difficult: the design problems and implemented algorithms are different, as well as the skill and experience of the software developer.
The development time is estimated for the software module that generates solutions, including knowledge acquisition, modeling, algorithm design, implementation and testing. The complexity of the generation module is measured by the number of parameters that a user can optionally specify as input: the degrees of freedom. The algorithm has to cope with changing input specifications and generate solutions that satisfy this input.
The required data points are hard to measure and scientific conclusions can only be drawn from a sufficient number of data points, obtained through controlled and repeatable experiments. For the data points I assume an error margin of 20 to 30%.
The functionality of six prototypes is given, as well as the development time of the generation module (not the complete application). After that, the development time is plotted against the number of degrees of freedom to illustrate the apparent lack of correlation.

4.1 Prototypes

One of the first prototypes is developed for compression spring design, with the graphical user interface as depicted in Figure 3. The interface shows the spring in three positions: relaxed, in first compression and second compression mode. Several input fields are visible where the user can optionally specify several geometric requirements and spring characteristics, in this case values for F1, F2, L2 and a maximum value for the external diameter. The algorithm generates a list of fully defined springs that meet the specifications, in this case a total number of 282 springs. Each solution is parametrically fully defined in terms of material, geometry and usage situation. Knowledge is used from an engineering handbook [19] and DIN standards. Development of the code that performs candidate generation took 12-16 weeks.
Prototypes with similar functionality as the compression spring designer are developed for spindle-drives, three types of springs and fiber-reinforced composites. The functionality of these prototypes is discussed briefly.
The spindle-drive designer enables the user to specify a desired motion and dynamic behavior of a manipulator that is positioned using a spindle-drive, powered by an electro motor and gear transmission. The software generates several combinations of electro motor, transmission and spindle. The algorithm takes into account the domains of dynamics, electronics and control systems and took approximately 14-18 weeks to develop.
Three spring designers are developed after the first version of Figure 3, for compression, extension and torsion springs. The development effort of the three systems was reducing due to the learning effect and re-use of code: approximately 10-14 weeks, 4-6 weeks and 3-5 weeks respectively.
The last prototype is the composite designer, which supports the design of fiber-reinforced composites. The user can specify which materials are allowed and information about the orientation and stacking of the plies. Different generation and optimization algorithms were implemented but the first algorithm (full search) took approximately 2 weeks.

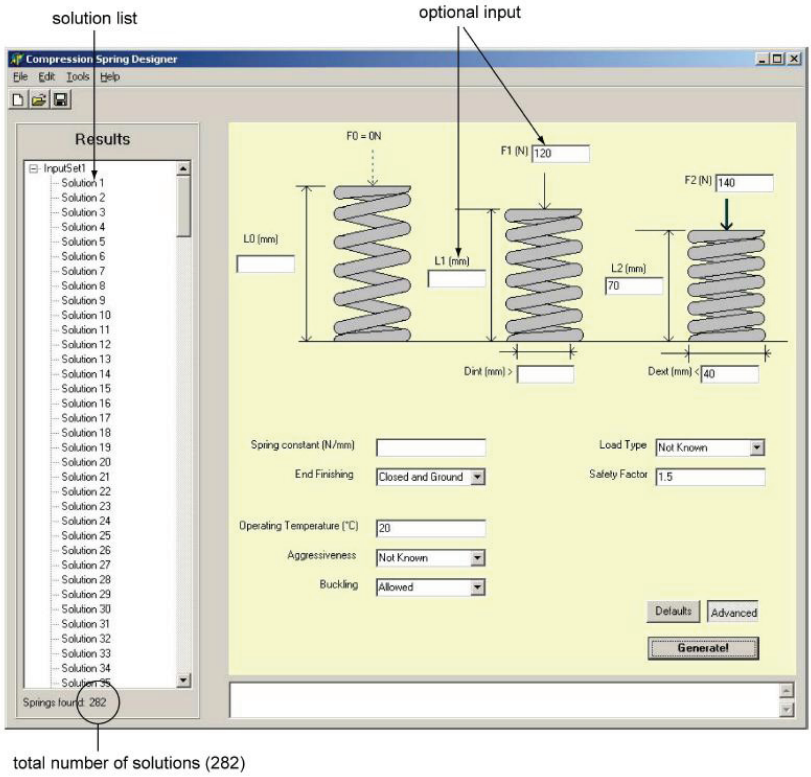

Figure 3: compression spring designer

4.2 Comparison

The relation between the development time and number of degrees of freedom is given in Figure 4. The figure illustrates the unpredictable nature of the development of the generation module: no apparent correlation exists between the development time and the complexity of the problem. Secondly, a development time of several months only for the design generation module of a mechanical spring seems out of proportion.

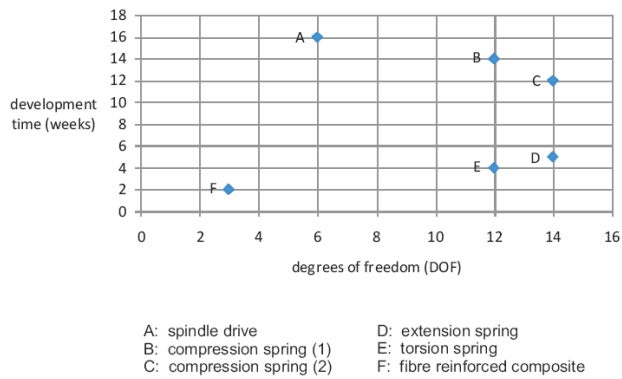

Figure 4: development time versus algorithm complexity

The scope of this paper is routine design problems that can be described parametrically. This means that the knowledge to solve the problem is available, tacitly, explicitly or a combination of both. The algorithms to automate such a problem are available. For instance the Role-Limiting Method [20] that expands, checks and repairs a partial solution. A similar algorithm from Constraint Satisfaction is backtracking [21], but a wider range or algorithms exist also from Constraint Solving and optimization research. The principle of having a generic algorithm that operates on a knowledge base is applicable for such algorithms and can be used to automate the intended scope, and one possibility is presented in [22].
Once a generic algorithm is prepared, the main difficulty to develop tools lies in the diversity of the design problems and the modeling of the knowledge. A systematic development method for design automation software should be indifferent to these aspects for its entire applicability scope. This is the domain of knowledge engineering. The following section discusses the domain of knowledge engineering to see if a readymade solution is available.

5 KNOWLEDGE ENGINEERING

Knowledge engineering (KE) is the process to design or develop a computational model of knowledge. KE involves activities of knowledge acquisition and representation [23], [24], both processes that have some distinct challenges.
Knowledge acquisition is the step during KE where design knowledge is made explicit. Schilstra [25] gives an overview of the development process of Expert Systems, and identifies several bottlenecks still persisting. These also include the tacit nature of expert knowledge, the challenges of knowledge extraction and the difficulty in modeling or representing the rules. In short, the well-known knowledge acquisition bottleneck [23].
Fensel [26] describes one of these difficulties by observing that design experts are experts in problem solving, not in explaining their solutions. The knowledge engineer therefore has to obtain thorough understanding of the problem at hand. Indeed, the book entitled “Fundamentals of Computer Aided-Engineering”, by Raphael and Smith, states that the most successful engineering knowledge systems have been created for situations where the engineer-developers were also well acquainted with the subject [27].
In general, KE is seen as an activity that requires understanding of both the computational aspects as well as the design case at hand. The knowledge engineer has choices to make during the representation of knowledge into rules. For the design of shape grammars, for instance, the ideal grammar should be comprehensive yet model only feasible designs [14]. This involves choices regarding the amount of rules: many relatively simple ones, or a single complex rule? And the level of parameterization of the problem: many parameters for good expressiveness, or fewer to for better computational performance. And how to describe the dependencies that occur between the rules?
The KE activity is usually done by people who *possess* the knowledge. This trend is seen in other research projects as well. The Knowledge Acquisition and Representation Language (KARL [26]) is aimed at knowledge acquisition from the *knowledge engineer* into a formal language. The MOKA project addresses the issues how to standardize and model design knowledge consistently, once it is gathered [28].
Studer et al. [20] reviews the principles and methods of knowledge engineering research. The paradigm shift from “direct transfer” of expert knowledge toward a “modeling activity” is discussed. The modeling activity of expert knowledge includes the process of acquiring tacit knowledge and make this explicit. When re-usable problem solving methods are used, the process of knowledge modeling is prescribed using the generic roles that knowledge can play. This “shell” approach is suggested and used for parametric design tasks. However, the inflexibility of the problem solving method and the connection to the real-life situations are said to remain a challenge.
The “classic” knowledge engineering is depicted in Figure 5, consisting of a knowledge source, a model and an algorithm. Model and algorithm are closely related because the algorithm more or less dictates the modeling expressiveness. A human interpretation of the knowledge source is required before it is modeled. This increases flexibility of algorithms and problem modeling, but also requires acquisition and understanding of the expert knowledge *as well as* understanding of the algorithmic possibilities and limitations.

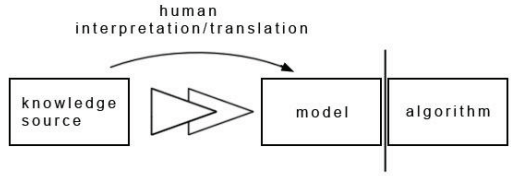

Figure 5: knowledge engineering

This paper suggests a closer integration between knowledge source and model, so that less human interpretation is required: a more prescribed translation of expert knowledge into modeling entities that are automated by a generic algorithm. A more prescriptive knowledge engineering activity is expected to reduce modeling effort at the cost of the applicability scope.

6 DEVELOPMENT METHOD

Figure 6 indicates the domains that are involved during development of design automation software. The continuous path through the domains, from left to right, indicates the development process.
The development process begins with a knowledge source, a design expert. The design processes where this expert is involved are first *decomposed* to gain overview and see where the software can be build. After selection of a suitable design process, the relevant knowledge is *acquired* and made explicit. Next, the knowledge is *modeled* into a format that is subsequently *automated* by an algorithm. When developing several software tools, the concept of *generic software development* is used to reduce the required effort. Finally, the *user interaction* is determined to offer the best interaction for the end-user.

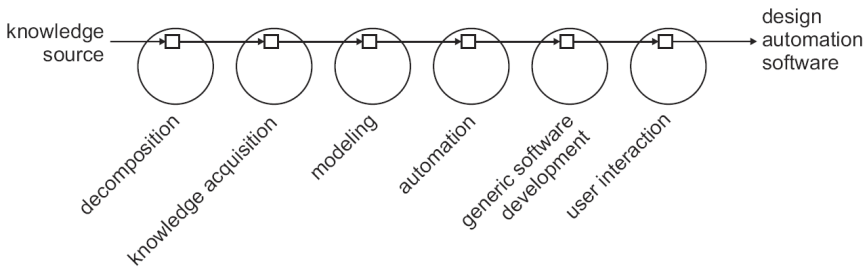

Figure 6. knowledge domains and the development process

The intended method translates a knowledge source into a model that can be automated by a known algorithm. The algorithm should be a generic one and applicable to an interestingly wide scope. The models of knowledge are defined in a standardized manner, and are automated without the need to modify the algorithm [22]. Prescriptive knowledge modeling requires standardization of the knowledge entities, but also consistency between the activities from knowledge source to knowledge model: do not acquire something the algorithm cannot automate.
A development method prescribes the KE activities from design process decomposition, knowledge acquisition, modeling and implementation. The goal is not only to make this process more predictable, but also for non-KE experts to be able to execute it. The method addresses three major questions:

1. what is relevant; what are the modeling boundaries;
2. how to acquire the computational model of the problem and the generative knowledge (knowledge acquisition);
3. how to automate the computational model: a generative algorithm.

The above mentioned questions should be answered simultaneously, rather than sequentially. The answers are found without the knowledge engineer becoming a design expert him/herself. The source of the knowledge states the required information during knowledge acquisition. Ideally, the answers of

the expert are implemented directly, without any intermediate translation or modeling by the knowledge engineer.
Some underlying principle is required to guarantee consistency between the algorithm and tacit expert knowledge. In our view, a human-like mechanism of design is best used as basis to derive the proposed method. Visser [29] discusses design from a cognitive point of view and describes the activity of design as an evolvement of representations until a complete design is created. The generation mechanism, or algorithm, can be an uninformed search. A relatively simple backtracking algorithm from constraint satisfaction can be used as baseline algorithm.
If a cognitive model is used as leading concept, the tacit knowledge resides in the mind of experts in a similar way as the automation algorithm will automate it. This reduces the need for intermediate knowledge engineering because the knowledge can go almost directly into the software.

7 CONCLUSION

Automation software for routine, parametric problems fit the context of today's and tomorrow's engineering design processes. Several domains play a role during the development of design automation software and each one *individually* seems mature enough to address a wide range of engineering design problems. Improvement of the development process is suggested by using a human-like knowledge model to reduce the modeling effort of expert knowledge. Subsequent automation can be done with an existing algorithm that is made generic for a scope of design problems. Consistency between the algorithmic capabilities and knowledge acquisition phase is maintained by using a human-like model of knowledge as underlying base.

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. I would furthermore like to thank Coen Cats for the development of the mass spring damper tool, and Robbie Woldendorp for development of the spring designer.

REFERENCES

[1] Schotborgh W.O. and Kokkeler F.G.M. and Tragter H. and van Houten F.J.A.M. and Tomiyama T. Towards a Generic Model of Smart Synthesis Tools, 2007, *proceedings of the CIRP Design Seminar 2007*
[2] Schotborgh W.O. Knowledge Engineering for Design Automation, 2009, PhD thesis, University of Twente
[3] Prahalad C.K. and Lieberthal K. The End of Corporate Imperialism, 2003, *Harvard Business Review*
[4] Henderson R.M. and Clark K.B. Architectural Innovation: the reconfiguration of existing product technologies and the failure of established firms, 1990, *Administrative Science Quarterly*
[5] Minderhoud S. and Fraser P. Shifting paradigms of product development in fast and dynamic markets, 2005, *Reliability Engineering and Systems Safety*
[6] Kuczmarski T.D. Managing new products: competing through excellence, in: Rao Artificial intelligence and expert systems applications in new product development- a survey, 1998, (Kluwer Academic Publishers)
[7] Rao S.S. and Nahm A. and Zhengzhong S. and Deng X. and Syamil A. Artificial intelligence and expert systems applications in new product development- a survey, 1999, Kluwer Academic Publishers
[8] Ahire S.L. and Dreyfus P. The impact of design management and process management on quality: and empirical investigation, 2000, *Journal of Operations Management*
[9] Rothwell R. Toward the Fifth-generation Innovation Process, 1994, *International Marketing Review*
[10] Green R. CAD Manager Survey 2004: Part 2, 2004, Cadalyst (www.cadalyst.com)
[11] Ullman D.G. Toward the ideal mechanical engineering design support system, 2002, *Research in Engineering Design*
[12] Simpson T.W. and Peplinksi J.D. and Koch P.N. and Allen J.K. Metamodels for Computer-based Engineering Design: survey and recommendations, 2001, *Engineering with Computers*
[13] Pawar K.S. and Riedel J.C.K.H. A survey of CAD use in the UK mechanical engineering

industry, 1996, *International Journal of Computer Applications in Technology*
[14] Antonsson E.K. and Cagan J. Formal Engineering Design Synthesis, 2001, (Cambridge University Press)
[15] Cagan J. and Campbell M.I. and Finger S. and Tomiyama T. A Framework for Computational Design Synthesis: Models and Applications, 2005, *Journal of Computing and Information Science in Engineering*
[16] Starling A.C. Performance-based computational synthesis of parametric mechanical systems, 2004, PhD thesis, Cambridge
[17] Hicks B.J. and Medland A.J. and Mullineux G. The representation and handling of constraints for the design, analysis, and optimization of high speed machinery, 2006, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 20: 313-328, Cambridge University Press
[18] Schotborgh W.O. and Kokkeler F.G.M. and Tragter H. and van Houten F.J.A.M. A Bottom-Up Approach for Automated Synthesis Tools in Engineering, 2006, *proceedings of International Design Conference 2006*
[19] Matek W. and Muhs D. and Wittel H. and Becker M. Machine onderdelen, 1996, (Academic Service Schoonhoven)
[20] Studer S. and Benjamins V.R. Knowledge Engineering: principles and methods, 1998, *Data & Knowledge Engineering*
[21] Kumar V. Algorithms for constraint satisfaction problems: a survey, 1992, *AI magazine*, 13(1): 32-44
[22] Schotborgh W.O. and Kokkeler F.G.M. and Tragter H. and Bomhoff M.J. and van Houten F.J.A.M. A Generic Synthesis Algorithm for Well-Defined Parametric Design, 2008, *proceedings of the18th CIRP Design Conference 2008*
[23] Feigenbaum E.A. How the "What" becomes the "How", 1996, *Communications in the ACM*
[24] Feigenbaum E.A. Some Challenges and Grand Challenges for Computational Intelligence, 2003, *Journal of the ACM*
[25] Schilstra K. Towards continuous knowledge engineering, 2003, PhD thesis
[26] Fensel D. The Knowledge Acquisition and Representation Language, KARL, 1995, (Kluwer Academic Publishers)
[27] Raphael B. and Smith I.F.C. Fundamentals of Computer-Aided Engineering, 2003, (Wiley)
[28] Stokes M. and MOKA consortium Managing Engineering Knowledge: MOKA, 2001, (Professional Engineering Publication)
[29] Visser W. Designing as construction of representations: A dynamic viewpoint in cognitive design research, 2005, *Human-Computer Interaction*, 21: 203-152

Contact: Wouter O. Schotborgh
University of Twente
Faculty of Mechanical Engineering
Laboratory of Design, Production and Management
Drienerlolaan 5, PO Box 217, 7500 AE, Enschede, The Netherlands
+31 (0) 53 489 2266
w.o.schotborgh@utwente.nl

Wouter Schotborgh studied mechanical engineering at the University of Twente and received his PhD in the area of knowledge engineering for design automation. His field of interest is design support tools for design experts.
Frans Kokkeler studied mechanical engineering at the University of Twente. He is a teacher and project leader of several projects relating to product development, software development and maintenance.
Hans Tragter is a researcher leading a group focused on the next generations of design tools. He holds degrees in architecture engineering as well as Computer science and Mechanical engineering (Twente), and has been involved in the development of CAD-systems 30+ years.
Fred van Houten is Head of the Laboratory of Design, Production and Management at the University of Twente. Relevant to this paper are his continuous endeavors to improve the quality and quantity of design support at front end of the process creation chain.