INTERNATIONAL DESIGN CONFERENCE - DESIGN 2012
Dubrovnik - Croatia, May 21 - 24, 2012.

AN INVESTIGATION INTO LIMITED INTEGRATION OF COMPUTATIONAL DESIGN SYNTHESIS IN COMMON DESIGN PRACTICE

F. Bolognini, J. M. Jauregui Becker and W. O. Schotborch

Keywords: computational design synthesis, generative tools, design methods, design process, common design practice, industry

1. Introduction

By Computational Design Synthesis (CDS) methods we intend all those techniques aimed at developing the synthesis phase of the design process through the algorithmic creation of designs. The aim is to leverage computational speed and depth of calculation to reduce the tedium of human designers and augment the process of searching the space of alternatives for preferred solutions [Cagan 2005].
The aim of computational synthesis methods is not just to find the best possible solution to a design problem: their emphasis is rather on creating design solutions that are, possibly, new design alternatives. Computational Synthesis as a method contrasts with traditional optimisation in that the goal of synthesis is more broadly to capture, emulate and/or utilise design decisions made by human designers during the creative process. Computational synthesis methods are complex products that do not just automate the optimisation of solutions. Their scope goes much beyond that, extending the concept of search of optimal designs with methods for creating solutions to propose to the optimisation. Also, some intelligence can be integrated in the way synthesis techniques offer solutions. Ultimately, the aim is to get machines to exhibit behaviour, which if done by humans, would be assumed to involve the use of intelligence [Simon 1969].
As things stand now for CDS development, the only possibility for designers seems to be working in partnership with computers, as complete automation of synthesis tasks is still far from being commonplace in industry [Bolognini 2009]. In most engineering fields, the use of generative design tools is at early stages and designers cannot rely on commercial packages to use in everyday practice. In the development of CDS methods, the involvement of creativity and other competencies that are unique to human mind, have been an obstacle. CDS methods also have scarcely attracted the attention of industry so far, to the point that it is essential at this stage to make a distinction between CDS methods and CDS tools (intending tools as complete of the shelf software packages ready to be used by designers). The fact that the diffusion of CDS in industry is so scarce is paradoxical, if we think of the advantages that CDS could bring. Among them the following:

- Better search of the design space in order to consider as many good solutions as possible
- Promotion of lateral thinking to aid innovation
- Insight into poorly understood tasks (due for example to interdependent parameters and constraints, coupled multidisciplinary performances, or emergent behaviours)
- Generation of multi-criteria design archives for investigating complex performance trade-offs
- Reduction of design time and cost
- Better direction of the overall creative effort.

Also, thinking of those rare design fields where CDS has been successfully integrated in design practice, it is difficult to foresee how the development of future complex designs in many engineering fields will be carried on without the support of these methods. Just to make an example, electric circuits have reached such level of complexity, that it would be practically impossible at this stage to design them without the support of CDS tools [Koza 2003]. From the investigation of some design fields like the electric circuits one, it is undeniable that CDS can be successfully integrated in design practice, but the reasons of these successes are at today unexplored. Factors like the design fields, the complexity of designs and variety of design tasks within the same field have an impact on success. Also, the scarce involvement of industry and non-academics designers in the development and research of CDS has been another possible cause of failure in the integration process (because of lack of synergy between academia and industry or just not the right players involved).
This paper explores the causes of the scarce development of CDS in industry and its integration in common design practice, based on direct experience of the authors and observation of the state of the art in the field. The paper also proposes a strategy to tackle the problem from different points of view: involving the right players of CDS development, exploring the right industry fields and design issues that could effectively benefit from the use of CDS, and finally looking into the development choices that could represent the basis for turning a CDS method into a CDS tool.

2. Background

For almost half a century computational design synthesis has been an active research field [Cagan 2005]. Already in 1969, Simon was providing the foundation of automated engineering synthesis methods with his paper 'The Science of Design', published in his book 'The Science of the Artificial' [Simon 1969]. Today, a wide range of problems may be solved through these techniques. The breadth of methods and results existing under the name of Computational Design Synthesis is so vast that it is almost impossible to review them all. A complete review of CDS methods developed in academia can be found in [Antonsson 2001] and [Chakrabarti 2002, 2011]. Some successful engineering field where CDS methods have been applied are structures, robotics, controls, MEMS and biology. In some cases these methods have been successfully integrated in design practice (e.g. topology optimisation methods for structural design and genetic programing methods for Printed Circuit Board design), while in some others their development and use lags behind (e.g. robotics and MEMS). The reason of this different outcome has never been clarified.
Probably the most successful field of application for computational synthesis methods is the electrical domain. Koza [Koza 2003] gives a complete review of all the devices synthesised using Genetic Programming, listing the patented circuit types that have been automatically generated, and showing how the electrical domain is also one of the most prolific fields of application for generative design tools. Structures have been the first field of application for synthesis techniques and a vast academic literature is available in this domain, also called structural topology optimisation. This field of application offered better commercial uptake than other engineering design fields. Especially in building design practice, topology synthesis has been investigated for many years with a vast employment of techniques, ranging from the use of discrete and continuous topology optimisation, to evolutionary based searches [Antonsson 2001]. Some of these methods also found successful, although modest, application in industry. Compared to the structural domain, fewer computational approaches for mechanical design synthesis exist. The difficulties found in this field relate to the fact that mechanical design covers a large and diverse range of problems where designs consist of many different types of interrelated functions and components. As a consequence, desired performance criteria cannot always be easily translated into quantifiable objectives. Mechanical applications approached using CDS have not found yet integration in design practice, if we exclude some examples of mechanical/kinematic motions and gear trains, like the gear synthesiser by [Starling at al 2005].
The other important point to stress is the fact that research in CDS development has been conducted so far exclusively at academic level. Some rare cases to be excluded are the project carried out by the Computational Science Division at NASA, GENRE, an evolutionary design system for evolving different types of antennas. Another successful example in this sense is the work of Lipson and Pollack in the electromechanical domain [Lipson et al. 2000]. Lipson and Pollack achieved the

automatic design and manufacture of robotic lifeforms by evolving electromechanical systems. These robots have now been developed by the Fraunhofer Institute, Germany. Other important works are those developed on compliant mechanisms for various industrial projects [Antonsson 2001]. The following section will try to explore possible causes of failure to develop CDS methods in some design fields and integrate them in common design practice.

3. Understanding the differences between successful and unsuccessful integration of CDS in design practice

From the previous section it appears evident that under certain conditions facilitate the set up of CDS methods and their employment in design practice. This happens for example when designs to be generated are formed by parts or components that are known or have a fixed shape (for example electric circuits), or when the design field is governed by rules that are known and understood. The opposite happens when rules defining a design field are unknown, or when a multiphysics environment generates coupling effects that can be unexpected or undiscovered. If the reasons why CDS can be successfully integrated in design practice are easily detectable, not the same can be said of all the causes that stop CDS from being employed at large scale. This section explores in details some of these causes.

Among the causes that are deemed unclear, there are those related to the unexplored connection between CDS and invention/creativity in design. Undoubtedly one of the goals of CDS is to boost innovation and help designers in the creative phase of the design process, but how the characteristics of these methods affect their ability to provide novel designs remains unknown. Figure 1 [Jauregui-Becker 2010] shows how routine, innovative and creative design is performed within different dimensions of design representations. The figure also shows that innovative design encompasses routine design, and creative design encompasses both innovative and routine design. From this perspective, understanding the rationales of creative design requires the previous understanding of innovative design, and likewise the understanding of innovative design requires the previous understanding of routine design. For the development of CDS tools, this means that before assisting creative and innovative design activities, a sound comprehension of the automation of routine design needs to be developed first. In this sense, it is expected that having libraries of automated routine design tasks will enable, after further research, the automation of more innovative and creative ones.

Although CDS in routine design has been broadly researched in academia [Clive et al. 1995], most methods have been developed for specific applications [Cagan et al. 2005] and remain confined in their task, without being evolved to creative design tools. For some reasons, the ground of innovative/creative design is handled with difficulty by CDS. The state of the art for generative design methods is far from giving real creative support to designers, to the point that even experts in the field do not associate creativity with computational synthesis. At the same time, and as pointed out by [Cagan et al. 2005], while it is true that advances in CDS have enabled the development of design automation methods for specific routine design problems, few methods describe how to do so from a general perspective. The lack of literature on systematic approaches to CDS development, even just oriented at unifying and comparing core techniques and case studies in the field, has also been considered one of the reasons why these methods have not been widely implemented in industry. The tight but unexplored relation between CDS and routine/innovative/creative design, as well as the unknown methodologies to structure CDS methods for these three different design purposes, deserve to be mentioned in this section. Although this topic is rightly mentioned here, it certainly needs to be investigated further in future works dedicated to its importance. Another cause of success for CDS methods to be considered is the use of specific optimisation methods in design. These techniques also impact the ability of CDS to approach innovative design (see for example the possibility to find novel solutions through the search of unconstrained space with stochastic methods [Bolognini 2009]). Because optimisation methods in Design Synthesis have not been the object of a systematic comparison yet, this topic also needs to be investigated at length in a dedicated work.

If the causes mentioned so far are more or less unexplored, others causes related to the development and use of the methods are known, and often object of research. Among them the following can be listed:

- The impossibility for CDS to scale-up to complex tasks
- The impossibility to readapt CDS methods to different design tasks or design fields: the development of methods for single design task or for testing purposes is common practice among developers
- The lack of multidisciplinary knowledge in CDS methods
- Software engineering problems as well as impossibility to interface CDS methods with external software for analysis and simulation.

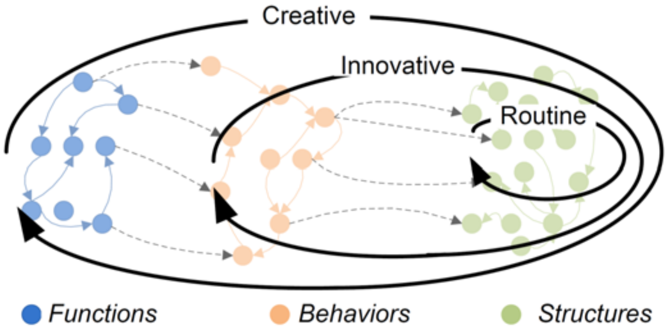

Figure 1. Bottom-up approach to computational synthesis research [Jauregui-Becker 2010]

At the basis of all the above mentioned causes is the underestimated approach of CDS developers to KE (Knowledge Engineering) theory and methods. The implementation of structured knowledge into a computational method is again an unexplored ground worth of mention in this work.
The following sections will investigate this last list of causes, analysing them from the three different points of view of developers, designers and industry. Developers are intended here as actual implementers of CDS methods. At this stage of advancement of CDS, they mostly identify with researchers, mostly academics, but could be identified in anybody that support their application to design processes (CDS community). Their background can range from engineering to programming. Designers are intended here as (potential) end-users of CDS methods, for whom CDS methods are developed). They can be single users or, most commonly, working for engineering companies. At this stage industry has to be intended as a employers at higher decisional levels that decide about strategic employment of CDS at industrial level.
An important issue to understand for spreading CDS technology in common design practice (and, ultimately, across industry) is the role of CDS development stakeholders (see Table 1).

Table 1. Summary possible causes of scarce use of CDS analysed from different perspectives

Cause	*Perspective*
Not clear difference between CDS and optimisation	Designers
Black box perception	Designers
Lack of expert knowledge integration into tools	Developers
CDS research projects usually missing designers' requirements	Developers
Lack of integration with PLM and CAE systems	Developers/Industry
Lack of systematic approach in developing CDS methods	Developers
No prescriptive method for CDS implementation	Developers
Knowledge Management not included into CDS methods	Developers
No clear design problem classifications and CDS application scope	Developers
Scarce interest in innovating the design process	Industry
Scarce understanding of links between innovation/efficiency and CDS	Industry
Incomplete communication of final goals of CDS	Developers

Research and development projects should therefore consider this structure in order to deliver technologies that can eventually be integrated into current design practice. Ideally, each stakeholder should be approached with different strategies, as their positions demand different requirements on

CDS development and use. Each stakeholder is somehow responsible for, or involved in, one or more of the causes of scarce spreading of CDS in the design community.

3.1 Lack of structured approaches in CDS development

Implementing a computational method of knowledge is a complex process, and for CDS in particular it is seen as something that requires understanding of both computational aspects as well as the design case at hand [Raphael and Smith 2003]. As in any KE (Knowledge Engineering) task, the major difficulty from a human source is the tacit nature of much expert knowledge and the fact that "design experts are experts in problem solving, not in explaining their solutions" [Fensel 1995]. And the largest cost element of building KBE systems occurs during the stages of gathering knowledge, extracting the useful parts and representing that knowledge in a structured way [Stokes 2001].
However, design problems in industry are quite different from academic cases in the sense that industrial problems often have tacit knowledge and the models are not explicitly known. The lack of the initial, explicit problem model could introduce the first task in CDS development for industry: knowledge engineering from a human source in a timely and efficient manner (Figure 2).

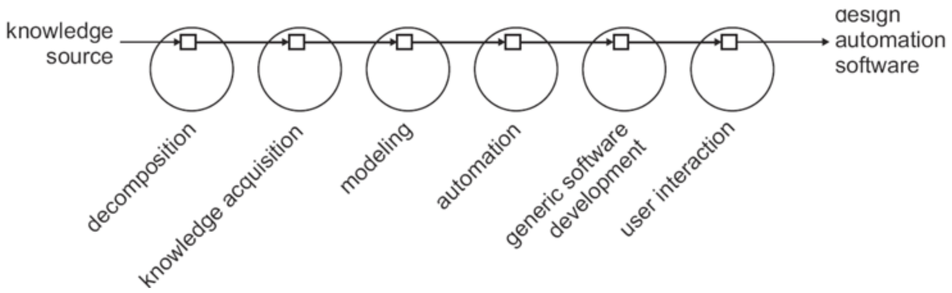

Figure 2. Development process for design software

The first step for doing so consists in decomposing a design process to define the model boundaries, followed by the acquisition of the required knowledge. Modelling and automation is the next step to develop a prototype tool. A possible bottleneck is 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 interviewing and gathering information from experts, or analysing textbooks, deciding upon system boundaries, quantity and content of knowledge rules and ensuring a maintainable knowledge base [Schotborch et al. 2012]. It has been noted by several experts that the creation of an initial model that describes a design problem, and derived from the tacit minds of designers, is both costly and difficult [Stokes 2001], [Cagan et al. 2005], [Fensel 1995]. The importance of the problem representation is also stressed as critical to the richness and sophistication of designs that are generated [Cagan et al. 2005]. And finally, proper software development and optimisation of user interaction are also needed in order to develop user-friendly tools in a commercially attractive manner.
In conclusion, to implement a CDS tool, developers need be able to follow a structured approach as the one indicated above. For this scope, they need to be experts in the design field as well as confident with algorithms and software development. In fact such professional figures are rare, especially because designers that become experts in a specific field rarely get involved in other sectors. On the other hand, it is difficult to think of developers as professionals that do not posses engineering knowledge, as this is only way to be aware of design issues. Considering these difficulties, it is understandable that developers and designers (intended as end-users of CDS products) should overlap, if not concur to the development of CDS tools. Other issues in CDS implementation that are connected to designers will be exposed in the following sections.

3.2 Implementation issues in CDS Development for future integration in design practice

The main justification for the limited use of computational synthesis in common design practice is that generative design methods are often restricted in their scope due to implementation issues. Therefore, this section explores the implementation structure of current CDS methods as an obstacle to the use of

CDS in common design practice. The section also explains the characteristics that CDS lack in order to be easily integrated into design practice and adopted at industrial level. Some fundamental or 'high level' functionalities of CDS have been identified [AAAI Spring Symposium on Computational Synthesis, 2003], that could allow their larger employment in common design tasks of ever growing complexity. Here are listed the two most important ones.
The first of these properties is the domain-independent nature of synthesis methods. Koza [Koza 2003] coined the expression 'high degree of routine-ness' to indicate the property of a generative design method has when relatively little human effort is required to get the method successfully handle new problems from different domains. Today, many current design automation paradigms focus on specific domains, making use of elaborate domain knowledge and domain-specific algorithms, and are typically limited to a single level optimisation. Multi-domain methods are instead universal methods that can address open-ended conceptual synthesis in a largely unconstrained design space. The ideal situation would be to develop generation strategies and algorithms that are uncoupled from specific problem domain knowledge. However, there should be the possibility of using domain knowledge to steer the solution generation process. An example of the latter is presented in [Schotborch et al. 2005].
The other property to be considered is scalability. This property allows expanding and upgrading of synthesis methods in order to handle increasingly difficult tasks. The main challenge is scaling synthesis algorithms so that they can achieve complex functionalities. Some characteristics are recognised to have an effect on scalability, as:

- Modularity: is the characteristic that allows the modification and expansion of methods through straightforward introduction of new features and functionalities.
- Regularity and hierarchy: are important aspects of making complex designs manageable and the synthesis process scalable.
- Abstraction and encapsulation of functionalities: for the synthesis process to scale, it is necessary to be able to abstract and encapsulate lower level design building components, so that the overall number of parameters remain constant as higher levels of complexity are reached.

In conclusion, good implementation criteria for CDS has been under investigation, although rarely taken into consideration, due to the fact that CDS developers may be expert in engineering design but not necessarily be skilled as software implementers.

3.3 Direct experience with designers

There is a large number of possible causes why the applied design community has not identified CDS technology as a possible resource for increasing the design process efficiency and effectiveness (thus, its quality and speed). However, there are three main concerns expressed by designers when being confronted with CDS. The first is the general perception that CDS is optimisation. Secondly, there is the tendency to mystify CDS as a virtual engineer. Designers' reaction is often one of not trusting CDS as a technology that can aid them in the process of designing new artefacts. CDS may be perceived as substitute of designers, not just as tools that may give them insight into tasks. For this reason, their expectations can be too high and the use of CDS may result disappointing, if not useless. On the one hand, the two first causes mentioned above are related to designers' lack of understanding of CDS. On the other hand, the third cause is related to the lack of flexibility that CDS offer when it comes to changing design requirements and feeding the system with different knowledge rules for design space exploration. As described in the previous section, a great deal of designing consists in changing and remodelling design requirements. Not offering these functionalities limits designers' ability to use CDS. Furthermore, this limitation also explains to some extends the perception on CDS methods as mere optimisation tools.

3.4 Involving industry: The big failure

In the lack of collaboration between developers (mostly academics) and industry in the growth of CDS tools, communication mistakes from the former in transmitting the final scope of CDS played an important role. Neither costs of developing CDS nor lack of competences in understanding and developing them seems to be in fact real obstacles to their use in industry. In the belief that potentials

and usefulness of CDS were not sufficiently stressed by developers, this section explores what is missing to get industry to give the necessary attention to CDS application in many sectors of design.
As mentioned in section 2, with counted exceptions, CDS research projects originate mainly at academic settings. CDS techniques are presently pushed into industry to aid solving design tasks that apparently can be solved without computational generative tools. In short, CDS offers a functionality that is not demanded by the industry. The general reaction to introducing CDS is: ‘Why do we need it, if we are already successful without it? However, there are closely related problems that industry is constantly experiencing, as the need for innovation and creativity. Another problem is how to manage knowledge in an environment of increasing complexity. To add more, designers nowadays tend to change jobs more often, producing sudden knowledge shortage in their current positions, which is difficult to quickly solve by hiring new staff. The continuous lack of expert designers produces less expertise in young designers that may need to be supported and guided in their decisions. From this contest, it looks like CDS is not expressing its potentials in terms of the real problems facing the industry, which results in a lack of interest.
CDS community should point to the three points identified above, in order to gain consideration at industrial level. Designers can solve complex design tasks without computational synthesis support, but the idea behind CDS theory is the possibility of designing novel and unexplored solutions, as well as gaining insight in tasks that are not easily understood. CDS advantages in design practice remain unknown or marginally explained to potential end-users, causing a wrong perception of their usefulness and the scarce tendency to include it in industrial plans.
In conclusion, it is again developers’ task to facilitate distribution of CDS theory and enable the adoption of CDS. At this stage, knowledge management in CDS development is not seen as a problem by developers and is not embedded in their research, although constituting an obstacle for research in the field. CDS community has not taken this knowledge management constraint into account yet, which leads to the development of techniques that apparently do not fit the existing design processes.

4. A recipe for integrating CDS in design practice

How to integrate CDS technology into design practice (and, consequently, at industrial level) is the end goal of this paper. In these sections we present a number of solutions for the CDS community to start acting in this direction. The first consideration originates from the fact that bigger interaction is needed among CDS stakeholders. At the state of the art, this is a fundamental step, especially for researchers and developers. These collaborations are presented here from the point of view of each player. Other considerations regarding KE techniques, and financial aspects of the research are also explained. By doing so, we expect to cover the most significant aspects required for CDS development and integration.

4.1 Involving the right stakeholders

As mentioned in previous sections, the potential players of the CDS community are of course developers, but also designers and industry. Each stakeholder should be approached with different strategies, as their positions and involvement demand different requirements on CDS development.
As seen in sections 3.1 and 3.2, if it is true that there are objective implementation issues that developers can solve on their own using expert knowledge and methods, it is also true that developers may not be designers or end-users of CDS tools themselves. For this reason, it may be a better strategy for developers to cover both positions. As this is not always the case, here comes the need of collaboration between developers and designers. It seems crucial that developers cannot avoid designers, which means that the latter have somehow to be part of CDS development process.
Design engineers, seen as CDS end users, should be integrated into CDS research projects as a source for determining the functional characteristics of CDS tools. One issue to address in this collaboration is concerned with understanding design challenges experienced by designers and the possibility to solve them using CDS techniques. These challenges can be categorised into three groups, namely non-routine design problems, complex problems and time-consuming problems. Another issue to address is the user interfaces of CDS tools. This goes beyond designing appealing software tools. It is important

to determine to which extent the end-users can make modifications on the solving methods and specify the general steps to follow when using CDS tools.
The second interaction would be between developers (mainly from academic background) and industry. As seen in previous paragraphs, developers should make clear how CDS can help solving a number of problems that industry has, but also many unexpected advantages that CDS employment could bring. At this stage (i.e. when CDS are not developed or presented as complete projects in industry yet) industry can be involved only at higher decisional levels. Managers should be put in the conditions to make consideration of what would be the impact of CDS on design processes at industrial level, on the opportunity to develop internal research on this topic, and which employers should be involved in these projects (for example a team of designers and software developers). In general, the advantages and the scope of CDS should be made clear by the CDS community, for a fruitful collaboration with industry and for starting up projects that could lead o successful applications. Another point that should be clarified is the fact that successful results can be obtained only through a combination of teamwork effort and investments.
The third point to consider here is the role of industry (intended as an organisation) and the collaboration among its own players (employees). This point is obviously to be considered only in the case that a company has actually accepted to include CDS in its design process and integrate it in its structure. Design Organizations (DO) is here defined as the whole human and physical infrastructure involved in a Product Development Process (PDP) in a company. As CDS technology has to be integrated into the existing management and developing process, understanding changes that such a technology would introduce into the existing schemes cannot be underestimated. Therefore, research is required to both support the integration of CDS into existing processes and support existing process interfaces. An example would be the question of how to integrate CDS into existing information management systems.

4.2 Knowledge engineering solutions

Synthesis is a knowledge intensive process. It is notable that different designers deliver different solutions to a same problem. The degree of differentiation between one designer’s solution and other designer’s solutions rests on two basic points: one is the level of openness of the problem and the second one is the diversity of the knowledge of each designer. In order to improve human design capabilities, both problem solving abilities and new knowledge structures have to be acquired. If computers are to (partially) take over the synthesis process for solving complex problems and delivering innovative solutions, they are required to exhibit somehow these two human cognitive properties. At least, the systems should be flexible enough to allow designers modify, expand and evaluate the knowledge bases being used for automatically generating solutions. Thus, a new shift in knowledge management systems would be required to deal with three main challenges: (1) allow designers decide which knowledge to use for both representing the initial design problem state and solving the design task, (2) use standardised digital knowledge building blocks (DKBB) that can differentiate between different levels of knowledge representation and (3) translate relevant knowledge into standardised digital knowledge building blocks.
For the first challenge, a solution approach is suggested by integrating 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. The mechanism should be closely connected to the human thought mechanisms to allow easy extraction of useful knowledge. The intended method translates knowledge into a model that can be automated by a known algorithm. An example of such a method is presented by [Schotborgh et al. 2012].
For the second challenge, a standardised representational language has to be developed. There is already a great diversity of representation schemes that can be found in CS design literature. While some representations are specific for one domain (e.g. super quadrics for shape design), other representation schemes, like grammars and parametric models, have proven to be of a more generic character. For example, design grammars have been successfully used in computational synthesis

methods for shapes design (e.g. [McCormack et al. 2004] and [Orsborn et al. 2006]), product design configurations (e.g. [Campbell et al. 2000]) and conceptual design (e.g. [Kurtoglu et al. 2008] and [Startling et al. 2005]). However, an integrated representation scheme is required to formally specify the relation between types of knowledge and types of representations. In order to define this language, a sound comprehension of design problems categories has to be elaborated. Dimensions to consider are both knowledge representations levels and mathematical models. Frameworks as the Function Behavior Structure [Zhang et al. 2005] and DF0 are examples of the former, while the structuring framework presented by [Jauregui-Becker et al. 2009] is an example of the later. Finally, the third challenge can be approached by developing prescriptive methods for design problem modeling and knowledge representation. Such types of methods for CDS would be analogous to discretization methods in Finite Element Method (FEM) as both serve to translate human representations into models suited for computational processes. Moreover, they can boost up CS technology by supporting preprocessing operations just as discretization methods have done for FEM [Cagan et al. 2005].

4.3 Business decision variables

As in most of human endeavours, business models enable durability and further development of technological advances. It is therefore important to consider some business variables at the moment of starting research projects involving all the stakeholders. Although this is not the central point of this paper, it is important to mention the most relevant business variables from the point of view of researchers. The first one would regard the licensing strategy (namely, open source of commercial). A second variable would be the scope of the tool to develop (single/multiple applications). A third would be the level of integration of CDS into existing CAE/CAE software, to support the existing processes as add-ins or to become a new piece of the puzzle as a standalone program. This would also imply whether to develop methods at kernel levels or at COTS levels. How to proceed with each one of these variables would depend on the characteristics of the challenges to be solved by CDS. From the point of view of industry, a business plan would of course consider the advantages (in terms of revenues) of using/developing CDS tools, as well as a measure of the advantages brought by innovation and speeding-up of the design process.

4.4 Possible scenarios

Other possible scenario in the development of CDS could be the introduction of new stakeholders: Computer Aided Engineering (CAE) and Computer Aided Design developers (intended as companies) as collaborators in CDS research or just as independent developers of CDS tools. This group would also be instrumental in developing tools to integrate in PLM tools. Such aspects as software maintenance, support and distribution would also be guaranteed.
Another point of view is that of determining industries' role in spreading CDS technology. Most research projects look in industrial partners potential tool developers. However, the experience of CAE has learned that this is not a feasible path for proving solid and robust solutions. Actually, when the first FEM ad CAD systems started to appear, the common practice was for each company to have its own CAD development department. As systems functionality increased in complexity, and developing times became too large, companies started to outsource these activities by buying COTS software. CDS could follow the same pattern of development and distribution. Companies are nowadays investing in research of technologies directly incorporated into their core business products. Unfortunately though, CDS does not belong to this category.

5. Conclusions

There is probably a long way to go before CDS tools will be perfectly integrated in common design practice. Developers cannot work on their own in this direction. In order to achieve this goal, developers, designers and industry will have to work in synergy. In each of these categories, it is important to find players that believe in CDS methods and understand their role in innovation and improvement of the design process. Also, possible new players and scenarios could unfold in the process of CDS integration, in the same way that happened for CAD tools. This work has introduced novel thoughts for the CDS community and also highlighted many more points to explore in the direction of this common objective.

References

Antonsson, E. K. & Cagan, J., "Formal Engineering Design Synthesis", Cambridge University Press, 2001

Bolognini, "An Integrates Simulation-based Generative Design Method for Microelectromechanical Systems", PhD Thesis, Cambridge, 2009

Cagan, J., Campbell, M. I., Finger, S. & Tomiyama, T., "A Framework for Computational Design Synthesis: Model and Applications. Journal of Computing and Information Science in Engineering", 5, 171-181, 2005

Campbell M.I., Cagan J. and Kotovsky K., "Agent-based synthesis of electromechanical design configurations", Journal of Mechanical Design, 122/61, 2000

Chakrabarti, A., "Engineering Design Synthesis", Springer-Verlag London, UK, 2002

Chakrabarti, A., Shea, K., Stone, R., Cagan, J., Campbell, M., Hernandez, N.V., Wood, K., 'Computer-based Design Synthesis Research: an Overview', Journal of Computing and Information Science in Engineering, Vol.11, 2011

Clive, L.D., 'Engineering Design: a Synthesis of Views', Cambridge Engineering Press, 1995

Fensel, D., 'The Knowledge Acquisition and Representation Language, KARL', Kluwer Academic Publishers, 1995

Jauregui-Becker, J.M, Tragter H. and van Houten, F.J.A.M., "Structure and models of artifactual routine design problems for computational synthesis". CIRP Journal of Manufacturing Science and Technology, 1(3):120-125, 2009

Jauregui-Becker, J.M., 'From How Much to How Many: Complexity Management in Routine Design Automation', Ph.D. Thesis, Faculty of EngineeringTechnology, University of Twente, 2010

Koza, J. R., Keane, M. A. & Streeter, M. J. "What's AI Done for Me Lately? Genetic Programming's Human Competitive Results", IEEE Intelligent Systems, 18, 25-31, 2003

Kurtoglu, T., Swantner, A. and Campbell, .I., 'Automating the Conceptual Design Process: From Black-box to Component Selection', Design Computing and Cognition '08, 553-572, 2008

Lipson, H. & Pollack, J.B., 'Automatic Design and Manufacture of Robotics Lifeforms', Nature, 406, 974-978 2000

McCormack, J. P., Cagan, J., and Vogel, C. M., "Speaking the Buick Language: Capturing, Understanding and Exploring Brand Identity with Shape Grammars", Design Studies, 25(1),1-29, 2004

Orsborn S, Cagan J, Pawlicki R and Smith R., "Creating cross-over vehicles: defining and combining vehicle classes using shape grammars", Artificial Intelligence in Engineering Design and Manufacturing, 20(3), 217–246, 2006

Raphael, B. and Smith, I.F.C., 'Fundamentals of Computer-Aided Engineering', Wiley, 2003

Schotborgh, W.O, Kokkeler, F.G.M, Tragter H., Bomhoff M.J. and van Houten, F.J.A.M., 'A Generic Synthesis Algorithm for Well-Defined Parametric Design', Proceedings of the18th CIRP Design Conference, 2008

Schotborgh,W.O, McMahon, E.C, Houten, F.J.A.M. van) "A knowledge acquisition method to model parametric engineering design processes", International Journal of Computer Aided Engineering and Technology, 4, 2012.

Simon, H. A., "The Science of the Artificial, Cambridge", MA, MIT Press, 1969

Startling A.C. and Shea K., 'A Parallel Grammar for Simulation-Driven Mechanical Design Synthesis', International Design Engineering Technical Conferences, Long Beach, 2(A), 427-426, 2005

Stokes, M., MOKA consortium 2001, 'Managing Engineering Knowledge: MOKA', Professional Engineering Publication, 2001

The Spring 2003 American Association for Artificial Intelligence, 'Computational Synthesis: from Basic Building Blocks to High Level Functionality', Standford, CA, 2003

Visser, W., "Designing as construction of Representations: A Dynamic Viewpoint in Cognitive Design Research", Human-Computer Interaction 21, 1, 103-152, 2006

Zhang, W.J., Lin, Y. and Sinha, N., 'On the Function-Behavior-Structure Model for Design'. Proceedings of Canadian Design Engineering Network conference 2005.

Dr. Francesca Bolognini
University of Cambridge
Trumpington Street
CB2 1PZ Cambridge, United Kingdom
Email: f.bolognini.03@cantab.net