Proceedings of the ASME 2012 International Design Engineering Technical Conferences &
Computers and Information in Engineering Conference
IDETC/CIE 2012
August 12-15, 2012, Chicago, IL, USA

DETC2012-71056

STEPS IN TRANSFORMING SHAPES GENERATED WITH GENERATIVE DESIGN INTO SIMULATION MODELS

Amir Hooshmand
Institute for Advanced Study
Technische Universität München
Lichtenbergstrasse 2a, D-85748
Garching, Germany
Email:
hooshmand@pe.mw.tum.de

Matthew I. Campbell
Department of Mechanical
Engineering
University of Texas at Austin
204 E. Dean Keeton St.
Austin, TX 78712-1591
Email: mc1@mail.utexas.edu

Kristina Shea
Engineering Design and
Computing Laboratory
Department of Mechanical and
Process Engineering
ETH Zürich, Zurich, Switzerland
Email: kshea@ethz.ch

ABSTRACT
This paper introduces a platform that combines shape grammars with conventional simulation and analysis methods. The premise of this combination is to create an approach to synthesizing optimal shapes considering criteria requiring heat transfer and stress analysis for their evaluation. The necessary mechanisms and issues for integrating shape grammars with standard simulation systems are described. The benefits, challenges and future outlook of this approach with regards to traditional design synthesis systems are explored. Further, possible future research projects to extend the work are presented.

Keywords: Computational Design Synthesis, Design Automation, Multi-physics Simulation, Finite Element Methods, Shape Grammar, Knowledge-Based Model.

1. INTRODUCTION
The significant amount of computing power and memory of today's computers has enabled the development of new methods and algorithms for diverse areas of application. However, the amount of design automation in different fields varies drastically. Computer aided tools for designing integrated circuits covers the whole design process from synthesis to simulation, analysis and optimization [1]. In mechanical engineering, Computer Aided Design (CAD) technologies are used mainly for analysis and representational purposes [2]. They do not typically cover synthesis and leave the most critical part of the conceptual design, i.e. finding a solution, to human designers [3, 4]. These tools mainly concentrate on analysis and optimization of specific details of a proposed solution [5].

The overall purpose of this study is to develop and implement a method to transform shapes created by a generative design method into a simulation model. The resulting platform provides automatic generation, evaluation and improvement of multiple design solutions. This paper is organized as follows. The second section contains background on topology optimization methods and the development of generative design synthesis systems. The third section presents the developed Computational Design Synthesis (CDS) platform integrating simulation. The fourth section is devoted to simulation friendly design synthesis. In this section, smooth integration of simulation with generative design through adequate development of design rules and design spaces is discussed. In section five integrating a knowledge-based model with shape grammars is discussed. Finally, in the last section conclusions and discussions are presented.

2. BACKGROUND

2.1. Topology Optimization
One of the most popular computational design synthesis approaches in engineering design involves topology optimization methods, which is based on using finite element methods (FEM) for the analysis, and various gradient-based optimization techniques [6]. For more than two decades, engineering designers have used shape and topology optimization methods for a wide range of structural design problems. These optimization methods are now being used successfully by other areas such as electro-magnetics, MEMS and fluids as well [6, 7]. Topology optimization is a mathematical approach that models a given fixed number of decision variables (cells or grids), and optimizes its objective

function (material layout) for a given set of boundary conditions and loads [6]. Numerical optimization methods have shown their efficiency in aiding the synthesis of engineering artifacts by generating many novel solutions [6] however they soon reach a limit in most of the engineering design problems.

One of the major limitations, which topology optimization methods in conceptual design are facing, is limited representation power; the synthesis process and design rules are dependent and integrated into the simulation model, the simulation model is often fixed for a given set of loads and boundary conditions. The reason to discuss topology optimization methods in this paper is not to reject it but to introduce a new perspective and show the abilities of generative design systems, such as shape grammars, in achieving more flexible design synthesis automation and optimization in the future.

2.2 Generative design synthesis systems

Due to the complexity of design problems to solve [8], which in turn comes from lack of knowledge about ill-structured design problems [9], a better understanding and a formal representation of the cognitive processes during different phases of design evolution are necessary to realize automated creative design [5]. In addition, to reach a high level of creativity, it is essential to go beyond the restrictions of already existing solutions and frames of reference [10] and extend the boundaries of the design search space. "Generative design systems are aimed at creating new design processes that produce spatially novel yet efficient and buildable designs through exploitation of current computing and manufacturing capabilities" [11]. Synthesis methods aim to assist designers in the creative phase of the design process and generate solutions that are novel and beyond a designer's own insight [12].

Through generative synthesis systems, designers are able to generate a large number of alternative solutions, to increase the quality of designs by increasing the chance to find a better design [13]. The designers must not only understand and decompose the design problem; they should also critically define the objectives, consider different decision drivers, and restrict the solution space in a way that richness of alternatives can be guaranteed [14]. After defining the design objectives, to use the maximum potential of generative systems, a design language for representing the system and a formalism for describing the generation process must be developed [14].

2.3 Shape Grammars

Grammars are strong tools for representing elements and their relationships in the design space [16]. Grammars capture large design spaces in a single formalism, and hence raise the design freedom [5]. Based on a set of pre-defined rules, grammars generate alternative design solutions [17]. Using a grammatical approach for different design application areas increased with the introduction of a shape grammar formalism by Stiny [18]. Shape grammars are mathematical tools for generating designs through a set of rules that can manipulate shapes and operate on them [18, 19, 20, and 21]. These rules are composed of a left hand side and right hand side shapes. The right-hand-side shapes are either a transformation of the left-hand-side or added/substituted shapes. Applying a rule means, recognizing the particular left hand side shape of the rule in the current working shape, and replacing it with the right hand side shape of the rule.

Although shape grammars are rooted in the architectural design literature [18, 19], and have been used to generate different designs such as Palladian villas [15], Prairie houses [22], and Chinese buildings [23], it has been also used for a variety of disciplines. This powerful generative mechanism has been employed in a wide range of design domains such as Coffee maker grammar [24], lathe grammar [25], truss grammar [26], Siza's Malagueira houses [27], MEMS resonator grammar [28], and also nontraditional areas like deriving cellular automata rule patterns by using shape grammars [29]. Important characteristics like direct handling of geometries, parametric geometry representation, and emergence are main reasons behind the success of shape grammars in other disciplines [28].

2.4 Shape grammar interpreters

Many scientists in different fields have developed computer tools to assist and automate in the development and application of shape grammar formalisms. A comprehensive list of these tools is presented by Gips [30], Chau et al. [31], and Hoisl and Shea [4]. In spite of the strong generative advantages of shape grammars and their application in a wide range of fields, development of shape grammar interpreters is still in its infancy. The development of a shape grammar interpreter is not a trivial task and faces many issues [32]. The developed shape grammar interpreters are limited to a certain algebra of designs [33]. These tools are mainly in 2D and those that are in 3D can work only with some certain limited rules. However, Hoisl developed one of the most powerful shape grammar interpreters in engineering design [4]. It allows users to create any arbitrary rule out of primitives in a user-friendly interactive environment [4]. The rules can be non-parametric and parametric and allow different transformations such as translation and rotation [4].

An important obstacle in the development of computational synthesis tools in engineering design is the difficulty in integrating the generation process with an efficient simulation package for guiding the generation process [12]. Indeed to find optimal designs it is of vital importance to automate the search process, which requires an adequate tool to facilitate the process of creating different variants as well as a suitable simulation tools to guide the search process according to evaluated engineering performance criteria.

Many scientists have tried to link shape grammars with a simulation model to evaluate the performance of the designs and guide the search process. Shea and Cagan have used FEM analysis to evaluate the performance of generated trusses and frames and guide the generation process [34]. Starling and Shea have used the behavioral modeling language Modelica to evaluate camera winding mechanism designs generated by the parallel grammar [35]. Bolognini et al. has coupled COMSOL multi-physics analysis with a synthesis method to generate MEMS [12]. However, all these examples are not general and have been developed only for one specific application, because coupling a simulation model robustly with design generation even for only one application is a complex task.

3. COMPUTATIONAL DESIGN SYNTHESIS (CDS) PLATFORM

To address the challenges in automatic shape generation and robust coupling with simulation methods, a new platform has been developed to increase the role of computers in generating alternative designs and exploring solution spaces for engineering problems. It introduces an approach that combines shape grammars with conventional simulation and analysis methods to provide guidance in design engineering according to evaluated engineering criteria. It uses an interactive, 3D shape grammar interpreter for describing solution spaces and generating design alternatives [4]. The shape grammar interpreter allows both interactive and automatic generation of alternatives. Finally, the evaluation results of proposed solutions with FE/CFD analysis will be used to guide the search process in a systematic way.

In engineering design, complex analyses (such as FE, CFD and thermal analysis) are required for calculating the engineering behavior of generated designs. Automatically integrating these analyses is a known challenge for engineers and designers. Aside from the difficulties mentioned and the large amount of time typically required for embedding external software packages in the automated synthesis process, doing this in a generic yet robust way is even more complex.

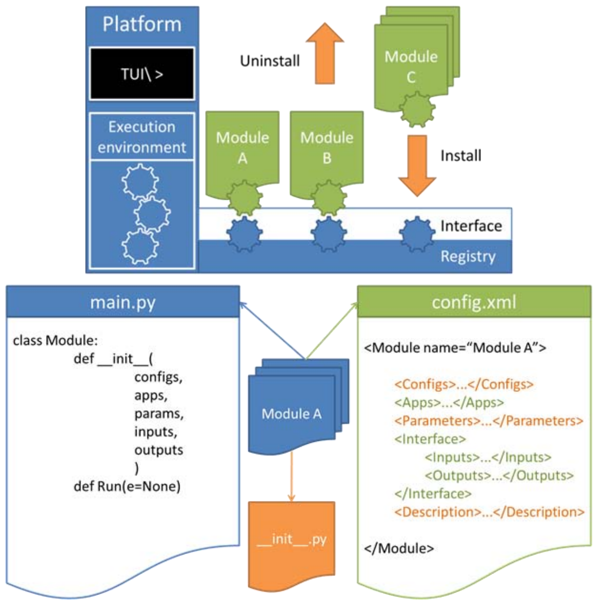
The major characteristic of the presented platform that distinguishes it from all other implementations is its generality. To reach this generality and flexibility, a module-based approach has been taken to develop the platform. The main characteristics of the developed module-based platform are as following:

- module management facilities (unified module registry)
- common interface for extensions
- modules execution environment
- file system data exchange, and
- multi-configuration system.

A module is a separate and interchangeable component of the platform, which accomplishes only one specific function. Fig. 1 illustrates the structure of the developed platform. According to a common interface among all modules, they must have a main.py file (Python file which contains class Module) and a config.xml file (metadata that describes module) (Fig. 1). The config.xml file contains all necessary information for running the module such as specific configurations of the module (e.g. type of the mesh in a preprocessor module), invoked applications (e.g. FE solver), environmental parameters, inputs of the module, and finally output results. All modules must be installed to the platform kernel registry. The registry is a subsystem of the platform that is responsible for the modules management. It consists of two layers:

- data layer: registry.xml file that stores records about installed modules and
- API layer: for modules management such as installing, uninstalling, and printing details.

The main idea behind the CDS platform is that it builds processes and executes them. A process is a sequence of modules, which is described by a process configuration file. Process configuration is an xml file that contains configurations of every module involved in the process, plus some additional information. It allows adjusting the process before every launch. The platform has three main module groups: generation, evaluation and guidance. To increase the flexibility of the platform, evaluation and guidance consist of different modules. These modules are indeed different types of methods to accomplish a task, for instance evaluation has one FE module for structural analysis and one CFD module for fluid mechanic analysis. A design process needs at least one module from each main module group.



```
class Module:
        def __init__(
                configs,
                apps,
                params,
                inputs,
                outputs
                )
        def Run(e=None)
```

```
<Module name="Module A">

    <Configs>...</Configs>
    <Apps>...</Apps>
    <Parameters>...</Parameters>
    <Interface>
        <Inputs>...</Inputs>
        <Outputs>...</Outputs>
    </Interface>
    <Description>...</Description>

</Module>
```

Figure 1. CDS PLATFORM: MODULE-BASED APPROACH.

3.1 Generation module

After benchmarking different available shape grammar interpreters (listed in Table 1 by Hoisl and Shea [4]), and considering the time and energy necessary to develop a new interpreter, this work is based on the developed shape grammar interpreter by Hoisl [4] as the platform's generator engine. The main criteria [4] to select this tool are as follows: support of 3D shapes, parametric shape grammars, transformations, shape types, definition/manipulation of rules, user friendly interface, and the capability to both execute shape rules automatically as well as interactively. These criteria have been discussed by Hoisl and Shea [4]. The shape grammar interpreter is based on a set of parameterized primitives. It provides an interactive environment to create 3D spatial grammar rules and their automatic application. The rules consist of two parts; Left hand side (LHS) and right hand side (RHS), in them different parametric or non-parametric geometric objects can be created and positioned in 3D space. Matching the left hand side of rules in a working shape and calculating the position and size of the objects in the right hand side rules is automatic [4].

3.2 Evaluation module

This module is focused on evaluating the quality of candidate designs according to defined criteria. As discussed, evaluating engineering behavior is a complex task and requires external tools for FE, CFD and thermal analysis. This has been achieved by integrating two preprocessors and solvers in the generation process; Salome preprocessor [36] and Code-Aster solver [37] for FE analysis and snappyHexMesh preprocessor [38] and OpenFOAM solver [38] for CFD and thermal analysis. Code-Aster is an Open Source software package for finite element analysis and numerical simulation of structural mechanics. It has been developed by a French company (EDF) as in-house software [37]. OpenFOAM is an open source CFD software that has been developed by the OpenFOAM Team at SGI Corp. OpenFOAM can be used for solving different problems in most areas of engineering and science from complex fluid flows involving chemical reactions, turbulence and heat transfer, to solid dynamics and electromagnetics [38]. SnappyHexMesh and Salome are two pre- and postprocessors that can be best coupled with these two solvers. By combining these two different preprocessors and solvers, multi-physics analysis of results will be possible. Many different criteria were considered for choosing these sets of solvers and preprocessors. The main criterion that has a direct effect on the synthesis process was the quality of results. Aside from the tests carried out by the developers of the software, many scientists in different disciplines also use these tools in their research [39, 40]. The second important reason for selecting these tools was the access to their source codes and possibility to customize them to the needs of the developed CDS platform. Open access to the codes was of vital importance for developing a general, generic CDS platform that unlike other implementations in this field is not restricted to any type of simulation or design. Due to a free licensing access to these analysis tools, the developed CDS platform can be offered to the CDS community for further development, use and investigation. It can provide a powerful base for scientists in the field of generative design synthesis to build their work on and help to reach the next generation of CAD tools exponentially faster.

3.3 Guidance module

In the generation process two decisions must be made; selecting a candidate shape to apply a rule and choosing a rule to proceed. Two main mechanisms have been developed to facilitate these two levels and guide the generation in a systematic way:

- Tree-search: in this mechanism the state of the current generation process (including all existing shapes) is stored in a tree structure. To apply the next rule (including choosing a candidate shape and a rule) one of the tree search algorithms such as depth-first or breadth-first can be used.
- Iterative mechanism: with help of this mechanism, only two solutions are saved, the current solution and the best solution, and the space can be traversed randomly, using directed search or by following gradients. In the CDS platform, a developed simulated annealing algorithm is responsible for the guiding the process.

These two mechanisms give the user the possibility to generate solutions either using tree search or guided with a stochastic optimization method, i.e. simulated annealing. Each mechanism has its positive and negative aspects that should be discussed extensively. For instance, the tree search mechanism increases the chance to reach the best solution but it is time consuming. The simulated annealing algorithm does not search the whole design space, but its efficiency to find optimally directed solutions (in designing frames and trusses) has been shown by Shea [34]. Aside from the possibility to apply different types of rules to change the topology, shape or size of the design, it is also possible to carry out a parametric optimization of intermediate or final solutions. For parametric optimization, the shape grammar interpreter is no longer required. Through this mechanism it is possible to combine traditional parametric optimization methods with generative design systems.

Fig. 2 shows the main mechanisms of the developed platform. As discussed, Evaluation, Guidance and Generation are the main modules. After applying a rule with either guidance mechanism, the shape grammar interpreter checks the validity of each candidate by evaluating defined constraints that could require simulation results. Based on some predefined criteria, the violation of constraints can be allowed to some extent using penalty functions. Those solutions that violate hard constraints are removed and the state of the design is updated and a new solution generated. Assessing different aspects of guidance mechanisms (tree-search and guided mechanisms) and comparing their results is not covered in the scope of this study and needs separate investigation.

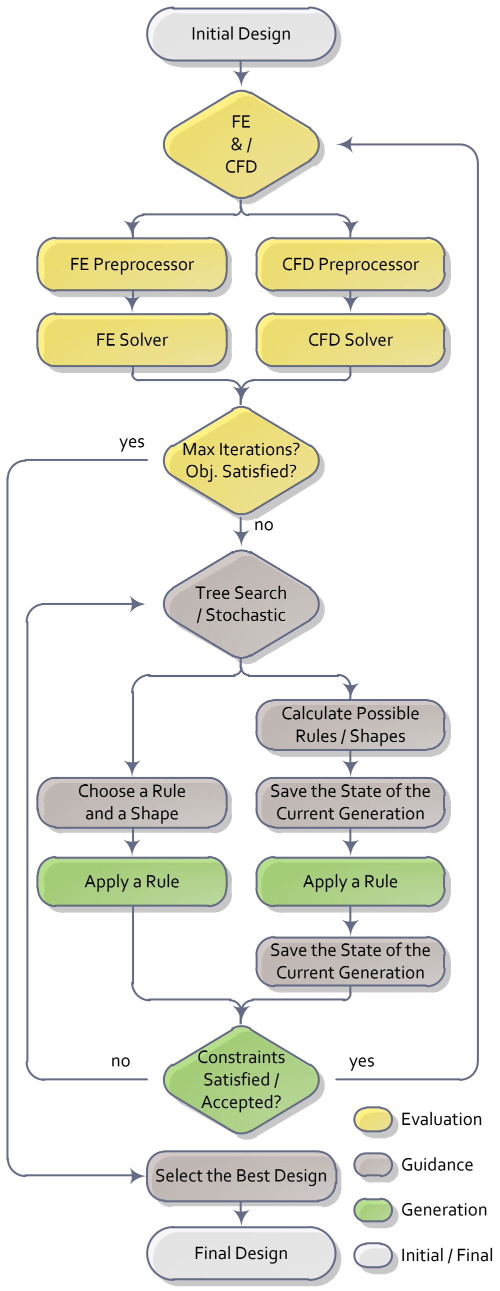


Figure 2. PLATFORM OVERVIEW.

Developing the CDS platform with all discussed modules required a large amount of programming effort in order to smoothly couple the simulation tools. In this paper we will now focus on the evaluation process and steps toward a simulation friendly shape grammar that is necessary to transform general generated shapes with generative design synthesis methods into valid simulation models.

4. SIMULATION FRIENDLY DESIGN SYNTHESIS

In the following, different mechanisms for overcoming issues when automatically transforming generated geometry into simulation-ready models such as geometry simplification, defining boundary conditions, loads and junctions are investigated. The following examples illustrate these issues using the example of generating alternative vehicle wheel rims, as found in [4], and present solutions.

4.1 Repairing and cleaning geometry before preprocessing

This step is the simplest one, because most of the available CAD/CAE software provide some functionalities to prepare a geometry for the preprocessing and creation the mesh. Providing geometry for preprocessing means removing those problems in the geometry that can affect the quality of the mesh. The most important issues are closing open contours, replacing coincident faces by one face (Fig. 3), suppressing small holes, and removing extra edges.

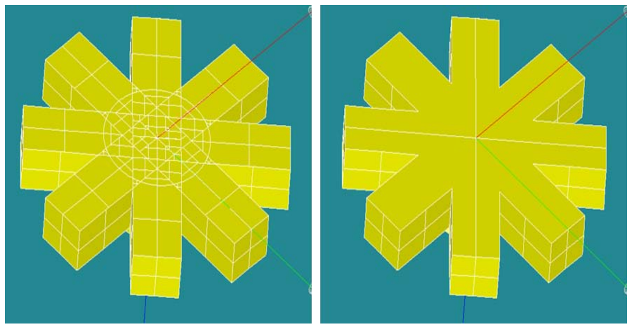


Figure 3. REPLACING COINCIDENT FACES BY ONE FACE.

4.2 Simulation friendly rules

This step is one of the most critical steps among all issues, since the design space is sensitive to changes in rules. Indeed, apart from considering simulation aspects in creating rules, “rule generation” is a difficult and time consuming task that needs a high level of expertise. To clarify, the subject is referred to the rim grammar and cooling fins grammar of Hoisl and Shea [4]. The rules for creating the rim solutions and the initial shape of the rim are illustrated in Fig. 4. By applying the illustrated rules in Fig. 4 on the initial shape (Fig.4 (e)), many different rim solutions can be generated (for example, see Fig. 5). These solutions have been created without any coupling with simulation, e.g. FE analysis. They have been created in an interactive way (not automatic) and visual inspection and engineering judgment have been the only criterion to guide the synthesis process.

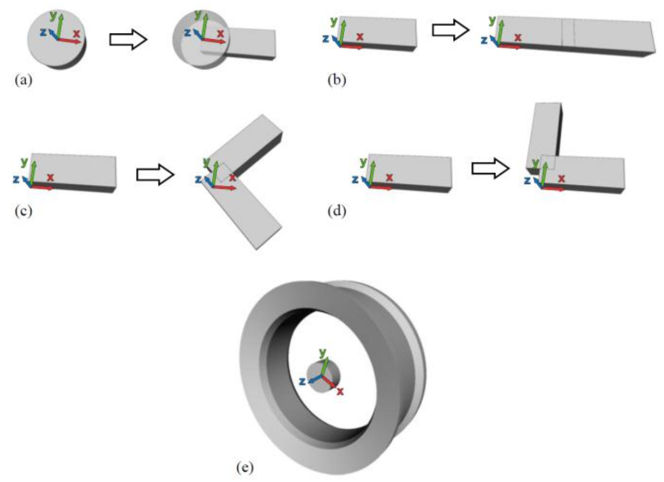


Figure 4. RULES FOR THE GENERATION OF THE RIM SPOKES (a)-(d) AND THE INITIAL SHAPE (e) [4].

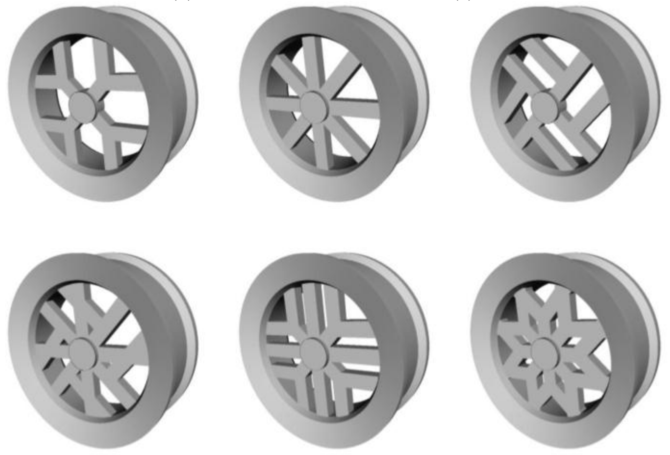


Figure 5. GENERATED RIM SOLUTIONS [4].

For an automatic generation of these rims with the CDS platform, results from a FE simulation are now used to guide the generation process. To accomplish the simulation, the outer surface of the rim and the inner cylinder of the initial design are used as boundary conditions respectively. Based on this assumption, rules (a), (c) and (d) are not simulation friendly. The reason can be seen in Fig. 6, which shows four intermediate designs after applying different sequences of rules. Two problems occur; first of all three generated rim solutions are not at all solvable in a FE solver, because there are no connections between the load carrier (inner cylinder) and boundary condition (outer surface of the rim). Second, even if the rim solutions could be solved with help of some provisional connections (narrow bars) comparing the intermediate designs generated is a critical task.

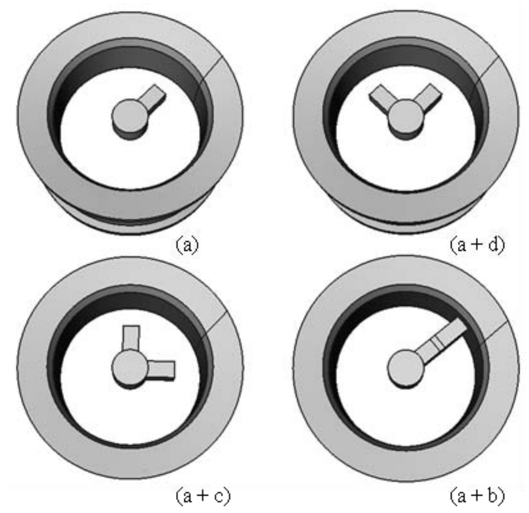


Figure 6. INTERMEDIATE RIM SOLUTIONS.

By considering the objective function of the simulation as minimum weight subject to strain as a constraint (or vice versa), after applying rule (a), (a+c) or (a+d) the objective function will be worse, because, these rules increase the weight of the rim without decreasing the amount of strain. Only by applying rule (b) after rule (a), the objective function is improved and the amount of strain will be decreased (Fig. 6 (a+b)). Rules (c) and (d) do not have this problem, when applying them on the outer half spokes. For instance, the sequence of applying rules (a+b) and finally applying rule (c) or (d) on the outer half spoke creates a simulation friendly design. This is in general an issue of initial designs where there are two or more disconnected parts.

Two strategies are suggested to solve the difficulty of evaluating and simulating intermediate solutions;

1. Simulating and evaluating the intermediate designs not after applying one rule but after a certain number of rule applications. This strategy solves the problem of instable intermediate designs to some extent and decreases the simulation time. However, evaluating the quality of individual rules is impossible.
2. Combining or changing the rules. For instance combining rule (a) and (b) will deliver desirable (in terms of simulation) results. One can generate completely new rules that produce always stable intermediate solutions.

The effect of both strategies on the final design might be negative, if the variation of the solutions is not considered. Therefore, considering simulation aspects when designing rules, must not affect the richness of the solutions generated. Therefore, changing the rule set including defining new rules can be critical. The next sub-section discusses how changing the design space can facilitate solving of the problem.

A closer study of the generated rim solutions in Fig. 5 reveals that the rims are symmetric to the center, which is logical. By changing the angle between the spokes of rule (c) and (d) from 90 degree to 180 degree, the illustrated rim solutions in Fig. 5 has been generated much faster. This change might affect the final solution but in case of symmetric designs, symmetric rules can help to solve the time issue of the simulation.

Generating simulation friendly rules is not limited to the above mentioned cases and can dramatically vary in different cases. These examples clarify the importance of considering simulation in designing rules.

4.3 Simulation friendly design spaces

Three issues in this section will be discussed, which should be considered in preparing the initial design space for simulation-driven synthesis. First, a proper definition of loads and boundary conditions is important for a simulation. In manual simulations, the designer chooses different points, lines, surfaces or volumes for applying loads or defining position restrictions. But for an automatic simulation of automatically generated designs, this step should be done during the synthesis or should be defined in the initial design. Both examples in Hoisl [4] have a proper initial design; the inner cylinder and outer surface of the rim are adequate for defining loads and boundary conditions. The cooling fins described also in [4] have a proper initial design since the inner cylinder can be used as a source of heating and the surrounding atmosphere as an outlet of energy.

Generating adequate intermediate designs in terms of simulation is the second issue. In the previous section, different strategies were suggested to solve the problem such as changing rules or the sequence of the simulation, but in this section changing the design space will be explored. By comparing the initial shapes of the cooling fins grammar (Fig. 7) and rim grammar (Fig. 4), it is seen that the initial shape of cooling fins grammar is a single block but in rim grammar it consists of two disconnected parts.

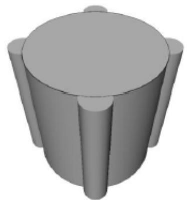


Figure 7. INITIAL SHAPE FOR THE COOLING FINS GRAMMAR [4].

As it is not always possible to use an initial shape that is simple and connected, a small trick is to use some provisional connections between different segments of the initial shape. As these connections exist for all evaluations, the results can be compared with each other. After finding the final designs, these provisional connections should be removed.

5. DYNAMIC KNOWLEDGE MANAGEMENT OF DESIGN SYNTHESIS PROCESS

The more knowledge that an engineer has about a given problem domain, the better will be the design decision-making and strategies that are employed. The strategy has a strong effect on the ability of the process to generate more alternatives. Strategy and knowledge are like two complementary aspects of the quality of generated alternatives. The generative ability of the strategy (generative design) used in this study has been discussed in the previous sections. In this section the focus lays on the knowledge and its application in the design process.

Due to the dynamic nature of the design process and evolution of designs during this process, it is difficult to develop a system with well-defined design variables and constraints to achieve the best solution via search mechanisms [41]. Hence, to find the solution of these ill-defined unstructured problems in large multidimensional design spaces [41], the predefined knowledge must be replaced with the newly discovered knowledge. Indeed, a key issue in developing new intelligent design automation approaches and tools is managing the dynamic information level of the design process. This management means confronting with three challenges: the ability to create new information (such as analysis and optimization results), the ability to make sense of this information (capable and intelligent algorithms for dynamic guidance of the process), and finally the ability to update information content of the design process (such as define reconfigurable rules). In the following all three aspects will be explored.

5.1 Creating information.

Analysis and optimization results are not the sole source for creating new knowledge in the design process. Their best solutions might have bad intermediate results. Fig. 8 shows two intermediate designs after applying twelve times the illustrated rules in Fig. 4. For the first intermediate design (i1), the sequence of rules is ten times (a), one (b) and one (c). Rule (c) has been applied on the result of applying rule (b) (the outer half spoke). Ten times applying rule (a) means creating the same spoke intentionally at the same place, therefore only one spoke consisting of three parts can be seen and the other 9 boxes are overlapping with them. The second intermediate design (i2) has been created with twelve times applying the same rules randomly. Surprisingly the results of the simulation show that the first rim (i1) has a better performance due to its weight, whereas the second wheel (i2) shows a much better generation process, because it can reach in a few steps more stable wheels. This example shows two problems in automated design processes. The first problem, which is important for the success of the synthesis, is the fact that a design that has a larger chance to reach a better final result (i1) shows worse simulation results. Consequently it might be filtered out through the guidance mechanisms. This problem shows the inability of simulation to prevent this issue. However, it can be facilitated to some extent by some guidance mechanisms; for example simulated annealing algorithm that accepts also worse solutions.

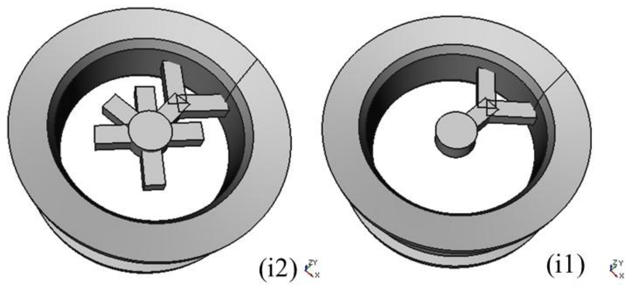


Figure 8. TWO INTERMEDIATE DESIGNS.

The second problem is that one can reach the i1 design only after applying three rules (a + b + c) and the other nine rule applications and consequently nine simulations are required. This is a huge problem especially when using one of the tree-search methods to guide the synthesis process. Fig. 9 shows possible children of intermediate designs for the first two levels of the tree. In the first level, it is possible to apply only one rule (a) and consequently reach one intermediate solution. At the second step, four rules (a, b, c and d) can be applied, therefore four children can be generated. From these four children three are novel and one of them (the result of applying rule a) is the same as the previous stage. In the third level 28 new solutions can be generated, from them nine are duplicates. In the next (fourth) level of the tree 280 children can be created. Interestingly, from these 280 designs, about 50% are duplicates. This number of possibilities increases exponentially when increasing in the number of generation levels. It must also be pointed out that the rim grammar with four rules is relatively a small design problem. So, it shows the importance of using extra mechanisms to prevent the generation of these useless intermediate designs and consequently useless simulations.

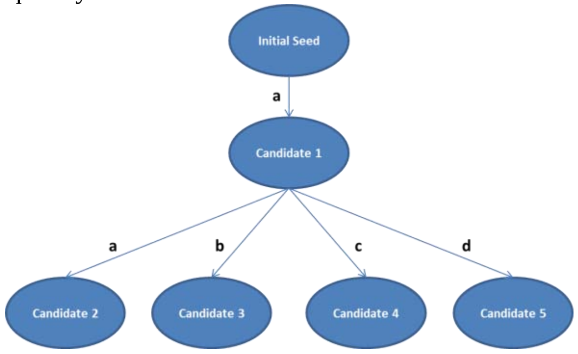


Figure 9. TREE STRUCTURE OF POSSIBLE CHILDERENS OF INTERMEDIATE DESIGNS.

5.2 Making sense of information (creating knowledge).

Grobler discusses the benefits and complementary properties of using knowledge-based models for supporting contextual information along with shape grammars for generating geometrical entities [43]. By saving the available contextual information of the design in a knowledge-model, it will be possible to query in this structured information, explore contextual aspects of the design and use this acquired knowledge for creating the next generation of designs.

Grobler remarks that computing and processing information in automatic and semi-automatic design of architectural artifacts faces with immense challenges [43]. However, these challenges are for industrial artifacts, due to more complicated geometries and implicit dependencies between different parts and disciplines, more difficult to handle. There are different types of representation forms for engineering knowledge, such as rule sets, procedures, features, parametric frames, and semantic networks [44]. In this research, the gained knowledge is provided to the shape grammars by means of a knowledge model. The model contains information regarding object types (primitives), size, place and transformation of objects, design parameters, context, and structure. Indeed, the rules can remain general and at the same time be customized in different contextual situations.

An example from previous sections can help to demonstrate the usability of the knowledge model for the design process. In section 4.3, it was discussed that the design space can be limited through specific rules. The issue was in the itemization of rules that in turn bring out the necessity to create more specific rules. Through the knowledge-based model it is possible to impede applying specific rules on specific objects with specific characteristics (in this study; spatial placement and transformation of the objects are checked), which do not add any value to the design. In this sense, the data will be used automatically to filter out the combination of those rules and objects that can lead to ill-structured designs for the next step. The issue of the Fig. 8 is another example that can be easily managed through adequate use of the knowledge model.

5.3 Updating knowledge (dynamic knowledge).

The dynamic characteristics that the model adds to the platform are explored in this section. Generative design mainly occurs in different phases [15]. Two main challenges of dynamic knowledge management of design synthesis processes -creating information and making sense of it- were discussed in the previous sections. In this section, different mechanisms to cope with the dynamic nature of the design process will be explored. As the amount of captured knowledge throughout the design process varies, the rules and specifications of the generative system should be in a dynamic state. One of the best solutions to reach a dynamic state in the design is to have reconfigurable rules (both shape grammar and knowledge-based model rules) based on the design knowledge. The rules should use the stored knowledge in the model and change their specifications. The dynamic knowledge of the design process can enhance the quality and fertility of the entire design process. The stored knowledge can be used also for other purposes, for example activating or deactivating different rules in different phases of design (such as topology rules, size rules, manufacturing rules), releasing/adding different constrains based on requirements.

6. CONCLUSIONS

This paper discussed the integration of shape grammars with conventional analysis and optimization methods to enable the generation of performance-driven shapes for engineering applications. To achieve this goal, a new platform has been developed that combines an existing interactive, 3D shape grammar interpreter [4] with conventional simulation and analysis methods.

This paper is devoted to investigating the necessary steps towards developing simulation-friendly shape grammars. Through simulation friendly shape grammars, the generation of adequate designs in terms of simulation is improved, which needs careful analysis of the design space produced by the design rules. This also reduced the generation of unreasonable intermediate designs that either violate a hard constraint or do not add any value to the design, such as creating redundant designs. Modifying the design rules and the initial design is beneficial in achieving simulation-friendly shape grammars but is not always possible. Future work includes further investigation of the requirements for developing simulation-friendly shape grammar rules considering a wider range of examples and validation of the robustness of the approach to automated simulation using benchmark topology optimization problems.

7. ACKNOWLEDGMENTS

With the support of the Technische Universität München – Institute for Advanced Study, funded by the German Excellence Initiative. The authors want to thank Frank Hoisl for his discussions on this work.

8. REFERENCES

[1]. Whitney, D. E., 1996, “Why Mechanical Design Cannot be like VLSI Design,” Research in Engineering Design, Springer-Verlag, London, Vol. 8, pp. 125-138.

[2]. Celani, M. G. C., 2002, “Beyond analysis and representation in CAD a new computational approach to design education,” Ph.D. Thesis, Massachusetts Institute of Technology, Department of Architecture.

[3]. Schotborgh, W. O., Tragter, H., Kokkeler, F. G. M., and van Houten, F. J. A. M., 2006, “A Bottom-Up Approach For Automated Synthesis Support In The Engineering Design Process Prototypes,” Proceedings, International Design Conference, Dubrovnik - Croatia, May 15 - 18, 2006.

[4]. Hoisl, F., Shea, K., 2011, “An Interactive, Visual Approach to Developing and Applying Parametric Three-Dimensional Spatial Grammars,” Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM), Vol. 25, Issue 4, pp. 333 – 356.

[5]. Alber, R., and Rudolph, S., 2002, “On a grammar-based design language that supports automated design generation and creativity,” Proceedings, IFIP WG5.2 Workshop on Knowledge Intensive CAD (KIC-5), Malta, Malta, July 23-25, 2002.

[6]. Bendsøe, M. P., Sigmund, O., 2003, “Topology optimization: theory, methods, and applications,” Springer, Berlin-Heidelberg, 370 pages.

[7]. Eschenauer, H. A., Olhoff, N., 2001, “Topology optimization of continuum structures: A review,” Appl. Mech. Rev., Vol. 54, Issue 4, 331-391.

[8]. Lewis, W., Weir, J., Field, B., 2001, “Strategies for solving complex design problems in engineering design,” In: Culley, S., Duffy, A., McMahon, C., Wallace, K., (eds) “Design research theories, methodologies and product modeling,” Proceedings, 13th International conference of engineering design, Professional Engineering Press, Glasgow, pp. 109–116.

[9]. Simon, H. A., 1970, “The structure of ill-structured problems,” Artificial Intelligent, Vol. 4, pp. 181–201.

[10]. Akin, O., and Akin, C., 1998, “On the Process of Creativity in Puzzles, Inventions, and Designs,” Automation and Construction, Vol. 7, pp. 123-138.

[11]. Shea, K., Aish, R., Gourtovaia, M., 2003, “Towards integrated performance-based generative design tools,” In: eCAADe 03, 21st Conference on Education in Computer Aided Architectural Design in Europe, Graz, Graz University of Technology, pp. 553-560.

[12]. Bolognini, F., Seshia, A. A., and Shea, K., 2007, “A Computational Design Synthesis Method for MEMS Using COMSOL,” Proceedings, the COMSOL Users Conference, Grenoble.

[13]. Heisserman, J., 1994, “Generative Geometric Design,” IEEE Computer Graphics and Applications, Vo. 14 (2), pp. 37-45.

[14]. El-Khaldi, M., 2007, “Automated Generative Synthesis Systems in Architectural Design,” 2006-2007 form Z Joint Study Journal, pp. 76-80.

[15]. Stiny, G., Mitchell, W., 1978, “The Palladian Grammar,” Environment and Planning B: Planning and Design, Vol. 5 (1), pp. 5-18.

[16]. Cagan, J., 2001, “Engineering Shape Grammars: Where We Have Been and Where We are Going”, in Formal Engineering Design Synthesis, ed. by Antonsson, E. K., Cagan, J., Cambridge University Press, New York, pp. 65–92.

[17]. Chase, S., 2002, “A model for user interaction in grammar-based design systems”, Automation in Construction, Vol. 11(2), pp. 161–172.

[18]. Stiny, G., 1980, ”Introduction to shape and shape grammars,” Environment and Planning B: Planning and Design, Vol. 7, pp. 343–351.

[19]. Stiny, G., and Gips, J., 1972, “Shape grammars and the generative specification of painting and sculpture,” Information Processing, Vol. 71, pp. 1460–1465.

[20]. Stiny, G., 1975, “Pictorial and Formal Aspects of Shape and Shape Grammars on Computer generation of Aesthetic Objects,” Birkhauser Verlag, Basel-Stuttgart.

[21]. Stiny, G., 1977, “Ice-ray: a note on the generation of Chinese lattice designs” Environment and Planning B: Planning and Design, Vol. 4, pp. 89-98.

[22]. Koning, H., Eizenburg, J., 1981, "The language of the prairie: Frank Lloyd Wright's prairie houses," Environment and Planning B: Planning and Design, Vol. 8, pp. 295-323.

[23]. Li, A., 2001, "A Shape Grammar for Teaching the Architectural Style of the Yingzao Fashi," PhD Thesis, Massachusetts Institute of Technology, Department of Architecture.

[24]. Agarwal, M., Cagan, J., 1998, "A blend of different tastes: the language of coffeemakers," Environment and Planning B: Planning and Design, Vol. 25, pp. 205-226.

[25]. Brown, K. N., McMahon, C. A., SimsWilliams, J. H., 1994, "A formal language for the design of manufacturable objects," Formal Design Methods for CAD (B-18), Eds. Gero, J. S., and Tyugu, E., North-Holland, Amsterdam, pp. 135-155.

[26]. Shea, K., Cagan, J., 1997, "Innovative dome design: applying geodesic patterns with shape annealing," Artificial Intelligence in Engineering Design, Analysis, and Manufacturing, Vol. 11 pp. 379-394.

[27]. Duarte, J. P., 2001, "Customizing mass housing: A discursive grammar for Siza's Malagueira houses," PhD Thesis, Cambridge (MA): Massachusetts Institute of Technology.

[28]. Agarwal, M., Cagan, J., Stiny, G., 2000, "A micro language: generating MEMS resonators by using a coupled form - function shape grammar," Environment and Planning B: Planning and Design, Vol. 27, pp. 615 – 626.

[29]. Crawley, E., Speller, T., Whitney, D., 2007, "Using shape grammar to derive cellular automata rule patterns," Complex Systems, Vol. 17, Number 79102.

[30]. Gips, J., 1999, "Computer implementation of shape grammars," In NSF/MIT Workshop on Shape Computation, Cambridge, USA.

[31]. Chau, H. H., Chen, X. J., McKay, A., and De Pennington, A., 2004, "Evaluation of a 3D Shape Grammar Implementation," Design Computing and Cognition 04, (Gero, J.S., Ed.), Kluwer Academic Publishers, Cambridge, USA, pp. 357- 376.

[32]. Jowers, I., 2010, "The construction of curved shapes," Environment and Planning B: Planning and Design, Vol. 37, pp. 42-58.

[33]. Stiny, G., 1991, "The algebras of design," Research in Engineering Design, Vol. 2, pp. 171 – 181.

[34]. Shea, K., Cagan, J., 1998, "Topology Design of Truss Structures by Shape Annealing," Proceedings, ASME Design Engineering Technical Conferences, Atlanta, GA, DETC98/DAC-5624, 1-11, September.

[35]. Starling, A. C., and Shea, K., 2005, "A Parallel Grammar for Simulation-Driven Mechanical Design Synthesis," Proceedings, ASME Design Engineering Technical Conferences, Long Beach, California, USA, September 24-28, 2005.

[36]. Salome: http://www.salome-platform.org.

[37]. Code-Aster, www.code-aster.org.

[38]. OpenFOAM, www.openfoam.com.

[39]. Silva, L. F. L. R., and Lage, P. L. C., 2011, "Development and implementation of a polydispersed multiphase flow model in OpenFOAM," Computers & Chemical Engineering, Vol. 35, Issue 12, pp. 2653-2666.

[40]. Lou, R., Pernot, J. P., Mikchevitch, A., and Véron, P., 2010, "Merging enriched Finite Element triangle meshes for fast prototyping of alternate solutions in the context of industrial maintenance," Computer-Aided Design, Vol. 42, Issue 8, pp. 670–681.

[41]. Lee, H. C., 2006, "The development of parametric shape grammars integrated with an interactive evolutionary system for supporting product design exploration," PhD Thesis, The Hong Kong Polytechnic University, School of Design.

[42]. Janssen, P., Frazer, J. H., and Tang, M. X., 2002, "Evolutionary Design Systems and Generative Processes," The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies, Vol. 16(2), pp. 119–128.

[43]. Grobler, F., Aksamija, A., Kim, H., Krishnamurti, R., Yue, K., and Hickerson, C., 2008, "Ontologies and Shape Grammars: Communication between Knowledge-Based and Generative System," Design Computing and Cognition DCC'08. J.S. Gero and A. Goel (eds), Springer, Atlanta, pp. 23-40.

[44]. Summers, J. D., Shah, J. J., 2002, "Empirical Studies For Evaluation And Investigation Of A New Knowledge Representation Structure In Design Automation," Proceedings, ASME Design Engineering Technical Conferences, DETC2002/CIE-34488, Montreal, Canada, September 29-October 2, 2002.

[45]. Orsborn, S., and Cagan, J., 2009, "Multiagent Shape Grammar Implementation: Automatically Generating Form Concepts According to a Preference Function," Journal of Mechanical Design (ASME), Vol. 131, Number 121007.