ICED13
INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED13
19-22 AUGUST 2013, SUNGKYUNKWAN UNIVERSITY, SEOUL, KOREA

CDS PLATFORM: A PLATFORM FOR MULTI-PHYSICS COMPUTATIONAL DESIGN SYNTHESIS

Amir HOOSHMAND (1), Marc SCHLAICH (1), Liliya BELAUS (1), Matthew CAMPBELL (2)
1: Technische Universität München, Germany; 2: University of Texas at Austin, United States of America

ABSTRACT
An important obstacle in the development of computational synthesis tools in engineering design is the difficulty in integrating the generation process with efficient simulation packages for evaluating candidates in a search process. The premise of this study is to develop and implement a platform to facilitate generative design systems in achieving more flexible design synthesis automation and optimization. This enables the designers to explore the abilities of generative design systems rather than coping with complexities of automatically integrating these analyses in the design process. The platform has been developed mainly based on open source software (OSS) to be offered to the Computational Design Synthesis (CDS) community for further development, use and investigation. Its modularity and programming based implementation provides a foundation for other researchers to build on and to achieve the next generation of CAD tools substantially faster.

Keywords: computational design synthesis, design automation, open source software, multi-physics simulation, finite element methods, shape grammar, graph grammar

Contact:
Amir Hooshmand
Technische Universität München
Institute for Advanced Study
Garching
85748
Germany
amir.hooshmand@pe.mw.tum.de

1 INTRODUCTION

In engineering design, complex analyses (such as FE, CFD and thermal analysis) are required for accurately predicting the engineering behavior of generated designs. Automatically integrating these analyses is a known challenge for engineers and designers. Aside from the inherent difficulties 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. Thus, an important obstacle in the development of computational synthesis tools in engineering design is the difficulty in integrating simulation packages with the generation process (Bolognini et al., 2007).

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 (Shea and Cagan, 1998). Starling and Shea have used the behavioral modeling language "Modelica" to evaluate camera winding mechanism designs generated by the parallel grammar (Starling and Shea, 2005). Bolognini et al. has coupled COMSOL multi-physics analysis with a synthesis method to generate MEMS (Bolognini et al., 2007). 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. Indeed the novelty of the presented platform lies in its generality.

The premise of this platform is to facilitate generative design systems, such as shape and graph grammars, in achieving more flexible design synthesis automation and optimization. This enables the scientist to explore the abilities of generative design systems rather than coping with complexities of automatically integrating these analyses in the design process. The CDS Platform uses several open source software, such as Salome (Salome 2012), Code Aster (Code-Aster 2012), Open Foam (OpenFOAM 2012), FreeCAD (FreeCAD 2012), SnappyHexMesh (OpenFOAM 2012) to perform a variety of multi-physics simulations. Unlike previous implementations, using open source software (OSS) in developing the platform enables us to offer the platform to the Computational Design Synthesis (CDS) community for further development, use and investigation. Its module and programming based implementation provides a powerful base for researchers to build their work on and help to reach the next generation of CAD tools faster.

This paper is organized as follows. The second section presents relevant open sourcing issues and its uses in product development. The third section presents the developed platform for the field of Computational Design Synthesis. In this section the system architecture and main applications are discussed. In the fourth section various application domains are illustrated and future outlooks are presented. Finally, the last section contains conclusions and discussions.

2 OPEN SOURCING

The appearance of open sourcing began in the 1960s, after the computer manufacturers decided to separate hardware and software, which provided the opportunity to develop software independently from the hardware (Hertel et al., 2003, Khanjani and Sulaiman, 2011a). The academic community, led by the University of California at Berkeley, defined a Unix-based Berkeley Software Distribution (BSD) that eventually lead to the Open Source Initiative (OSI; 1998). The OSI defined open source as code that: is distributed freely, can be modified freely, and is accessible to a large number of developers through the Internet. According to Deshpande and Riehle (2008), the growth in open source doubles almost every year in term of the projects and the number of lines of code. There are several open source software properties that have advantages when used for product development (Ruffin and Ebert, 2004). Open source projects have a longer lifespan, heed standardized interfaces and are easier to integrate with other software tools.

3 CDS PLATFORM

To address the challenges in design synthesis 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 and graph grammars with conventional simulation and analysis methods to provide guidance in design engineering according to evaluated engineering criteria. The major characteristic of the presented

platform that distinguishes it from all other implementations is its generality. To achieve this generality and flexibility, a programming and module-based approach has been adopted in developing the platform.

3.1 A module / programming based platform

Due to the complexity and the wide range of possible applications it is not feasible to develop a software that can include all possible simulation scenarios. Instead, the CDS Platform takes the approach that enables the users to develop the corresponding synthesis processes on their own. This is supported in the form of a programming framework in Python. This means it provides an interface so the user can implement the design synthesis process. By this approach multiple forms of control flow are supported and various ways of creating macros are provided, for example in form of functions or object-orientation.

3.2 System Architecture

The platform architecture is inspired on the general synthesis cycle illustrated in Figure 1. For every stage of the process, one most define the corresponding modules. Additionally, some functionality is necessary to convert the output data of each step to fit the requirements of the next step. Modules have access to any information created or available in other stages, because the information is stored central in text format.

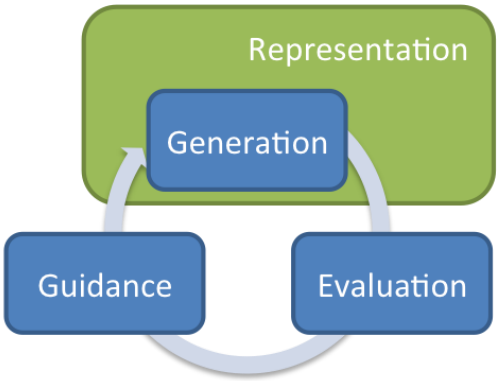

Figure 1. General synthesis process, modified from (Cagan et al., 2005)

The components of a CDS framework are presented in Figure 2. For each step there are usually multiple modules that can provide the required functionality, so the platform is not limited to single tools or processes. In the following subsections, each of the seven module-types is briefly described.

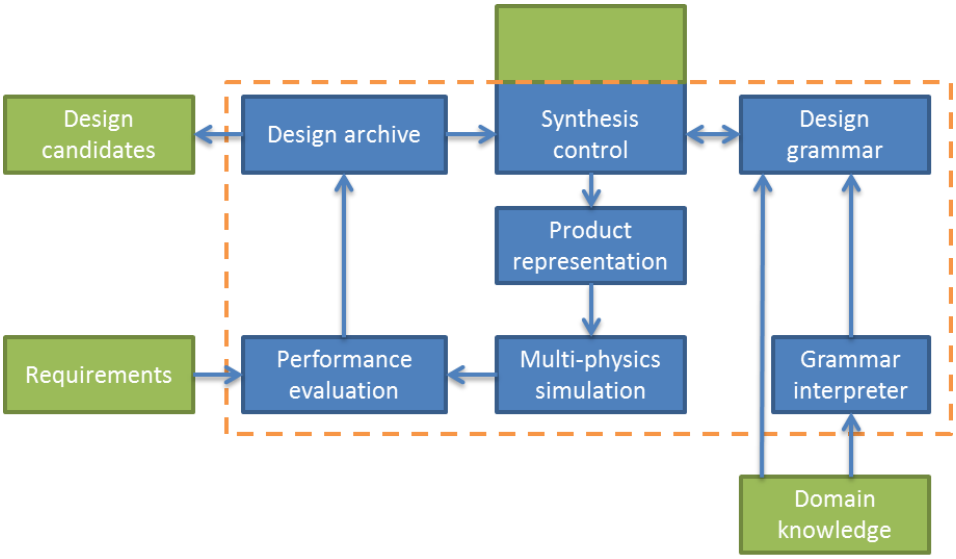

Figure 2. Components of a CDS framework modified from Helms et al., (2009)

For **multi-physics simulation**, the platform integrates Code-Aster (Code-Aster 2012) for Finite Element analysis (FE) and OpenFOAM (OpenFOAM 2012) for Computational Fluid Dynamic (CFD) and thermal analysis (as part of the evaluation module). The necessary converters and preprocessors needed for these tools (Salome and snappyHexMesh) are integrated in the platform as well. By combining these sets of different preprocessors and solvers, multi-physics analysis of candidate solutions is 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 thousands of tests, which have been carried on through developers of the software, a brief search in the literature revealed that many researchers in different disciplines have used these tools to accomplish their scientific research (such as Silva and Lage, 2011 and Lou et al., 2010). The second important reason for selecting these tools was their open-source nature that facilitated the integration with the developed CDS Platform. Open access to the source code was of vital importance for developing a 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.

The **performance evaluation** itself is realized by gathering information from the whole synthesis process (mainly from simulation) and combining it into a single objective value by the means of an *aggregating function* that does a weighted addition of all collected data (Wang et al., 2008).

The **synthesis control** is mainly a task of the user due to the nature of a programming framework. But the platform provides an easy access to a lot of generation approaches like optimization, search trees and knowledge-based processing. As the user influences the control flow, he can easily integrate additional approaches like Genetic Algorithm (GA).

As a simple **grammar interpreter**, the CDS Platform can access a shape grammar interpreter that is introduced by Hoisl and Shea (2011). For the **representation** the *FreeCAD* file format is used in this case. To support more sophisticated designs and grammars, the platform will be integrated GraphSynth as a graph grammar interpreter developed by Campbell (2006). It has been used by researchers such as Kurtoglu et al. (2010) and Rai et al. (2011). As the results are **represented** in graphs, they must be transformed into shapes. This task is carried out through integrating commercial or open-source CAD kernels (e.g. Parasolid, ACIS or OpenCasCade). This kernel is not open source.

3.3 Integrated tools

3.3.1 Multi-physics simulation tools

Code-Aster is an Open Source software package for finite element analysis and numerical simulation of structural mechanics and Civil and Structural Engineering. It has been developed by a French company (EDF) as an "in-house" software (Code-Aster, 2012). Code-Aster was released as free software under terms of the GNU GPL in 2001. Code-Aster is mainly a solver for mechanics, based on the theory of the finite elements (FE). This tool covers a large range of applications: 3D thermal analysis and mechanical analysis in statics and dynamics, for machines, pressure vessels and civil engineering structures. Beyond the standard functionalities of the software for solid mechanics, Code-Aster compiles specific research in various fields: fatigue, fracture, contact stresses, geo-materials, porous media, and multi-physics coupling. The **Salome Platform** can be best coupled with the Code-Aster solver to effectively preprocess the geometries. Salome is an open source software platform which has been started in 2001 and distributed with the GNU LGPL license. It provides a generic pre- and post-processing tool for numerical solvers. 3D solid shapes are transformed into tetrahedron or hexahedron meshes in the mesh module to be prepared for finite elements analysis. Post-processing module of the Salome allows importing and analyzing calculation results generated by CAE solvers (Salome, 2012).

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 areas of engineering and science from complex fluid flows involving chemical reactions, turbulence and heat transfer, to solid dynamics and electromagnetics (OpenFOAM, 2012). The latest application of OpenFOAM also includes stress analysis, large strain analysis and magneto-hydrodynamic flows (Karac, 2003). It has a large user base across both commercial and academic organizations. OpenFOAM includes tools for meshing, notably snappyHexMesh, a parallelized mesher for complex CAD geometries, and for pre- and post-

processing. **SnappyHexMesh** generates 3D hexahedra meshes from a triangulated surface geometry in STL format (Ribes and Caremoli, 2007).

3.3.2 Synthesis control tools

In the generation process defined by a grammar a decision must be made among options which include a location within the candidate (e.g. a subshape or subgraph) and a rule that modifies that location. Two main mechanisms have been developed to guide the generation in a systematic way:

- Tree-search: the state of the current generation process (including all existing candidates) 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 is used.
- Iterative mechanism: only two solutions are saved, the current solution and the best solution, and the space can be traversed randomly or by following gradients. To date, simulated annealing algorithm has been developed for guiding the process.

These two mechanisms have 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. An iterative mechanism like 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 (Shea and Cagan, 1998). 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 requires further investigation. **SciPy** is another open source optimization toolbox which has been integrated in the platform to be used in the guidance process (SciPy, 2012).

3.3.3 Grammar interpreters

A 3D shape grammar interpreter developed by Hoisl and Shea (2011) has been fully integrated in the platform and the integration of a Graph Grammar Interpreter (GraphSynth) developed by Campbell (2006) is under development. The grammar interpreters are used for describing solution spaces and generating design alternatives. They allow both interactive and automatic generation of alternatives.

The main criteria to select the 3D shape grammar interpreter 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 (2011). The shape grammar interpreter has been developed within the FreeCAD environment. FreeCAD is an open source software distributed under GNU GPL and LGPL license that supports parametric 3D building of volumetric models (FreeCAD, 2012). The software supports several import/export document formats. To make drawing in FreeCAD convenient, scripting in Python is added to the software that allows users to create and modify geometries effectively.

GraphSynth is a unique research software for creating, editing, displaying, and manipulating generative grammars. This framework stores graphs, rules and rulesets in an XML file format. This allows automatic search for creative, optimal or targeted solutions. Additionally, it is able to perform various graph transformations such as the double-pushout method and free-arc embedding; these two together cover nearly all types of required graph transformations (Campbell, 2006). One of the most important characteristics of the GraphSynth is its expandability; through additional C# functions (compiled on-the-fly by GraphSynth) any capability can be added to the rules and rulesets.

3.4 Data exchange

The data flow in the platform is highly dependent on the use case or synthesis process because the user is mainly responsible for the control flow. But there is a common pattern in data flow that is in use while communicating with the integrated tools (for each module). The general approach to communicate with any of the integrated tools is to export a file, which is used as input to the subsequent module. As next, the application is called by its command line interface to generate the necessary output. This output needs to be parsed (usually by the means of regular expressions) so it can be passed backed to the platform. This pattern is generic, platform and language agnostic, but it has its drawbacks primarily in maintainability, because the input and output specifications of the integrated tools could change in newer versions and it is harder to debug it than a software using a programming interface. Nevertheless it is the only option, as most of the tools do not provide an API in Python.

4 APPLICATIONS

Applicability of the platform is directly dependent to the grammar interpreter abilities to generate new design solutions. The grammar interpreter defines not only the type of the problem which can be solved but the richness and quality of the solutions connected to it. In the following sub-sections, the applications are discussed that have been built using CDS Platform.

4.1 Shape synthesis for axisymmetric problems

The first grammar interpreter that was integrated in the platform was a shape grammar interpreter, developed by Hoisl and Shea (2011). They have illustrated various design problems for their 3D shape grammar interpreter including cooling fins grammar and wheel rims grammar. Through integrating the grammar interpreter in the platform, both of these problems are solvable with the platform and have been extensively discussed by the author in Hooshmand et al. (2012). In Figure 3, different components, which have been used in the CDS Platform for wheel rim synthesis, are illustrated. Although the generated design solutions with the shape grammar interpreter are novel concerning topological aspects, due to the nature of the interpreter –which relies only upon primitive shapes such as boxes and cones for the representation–, the results are not industrially applicable or valuable. However, the generated results by the shape grammar interpreter point to the future possibilities in the field of CDS. Specially, through using the platform in the future projects, the researchers will not have to struggle with integrating complex simulation models in their implementations. While currently limited in terms of the types of shapes that can be defined, future improvements in shape grammars would lead to more complex shapes.

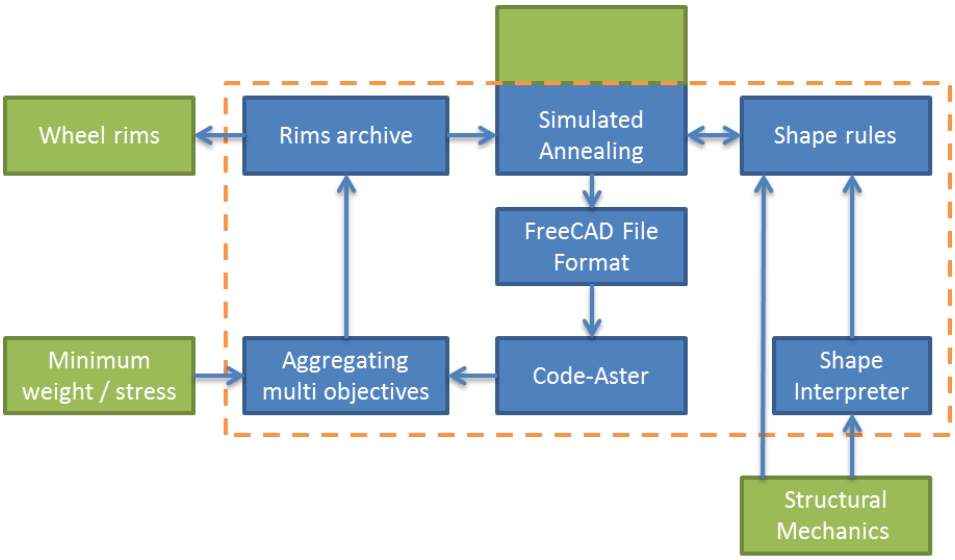

Figure 3. CDS Platform components for wheel rim synthesis

4.2 Fluid channel synthesis

Optimization of fluid channels is an essential topic in designing microfluidic devices (Andreasen et al. 2008; Vangelooven et al. 2010). The goal is mainly to find an optimal topology for the fluid subdomains along with an optimal shape of channels (Liu et al., 2011). Borrvall and Petersson (2003) used for the first time topology optimization for solving fluid problems in stokes flow. Since then, many scientists have used various grid-based topology optimization methods to solve fluid layout 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 (Hooshmand et al., 2012). The process of solving a layout problem, even for simple 2D is very time consuming.

Through combining the generative abilities of GraphSynth with the CDS Platform, we are overcoming the limitations of current methods. The effectiveness of the proposed method is checked by solving a variety of available test problems and comparing them with those found in the literature. Furthermore by solving very complex large scale problems the robustness and effectiveness of the method is tested.

GraphSynth creates the topology design and OpenFOAM evaluates the candidates. Figure 4 shows the CDS Platform components for solving fluid channel layout synthesis. As can be seen the components are almost fully different from those illustrated in the Figure 3 for solving wheel rim synthesis, which shows the flexibility of the CDS Platform.

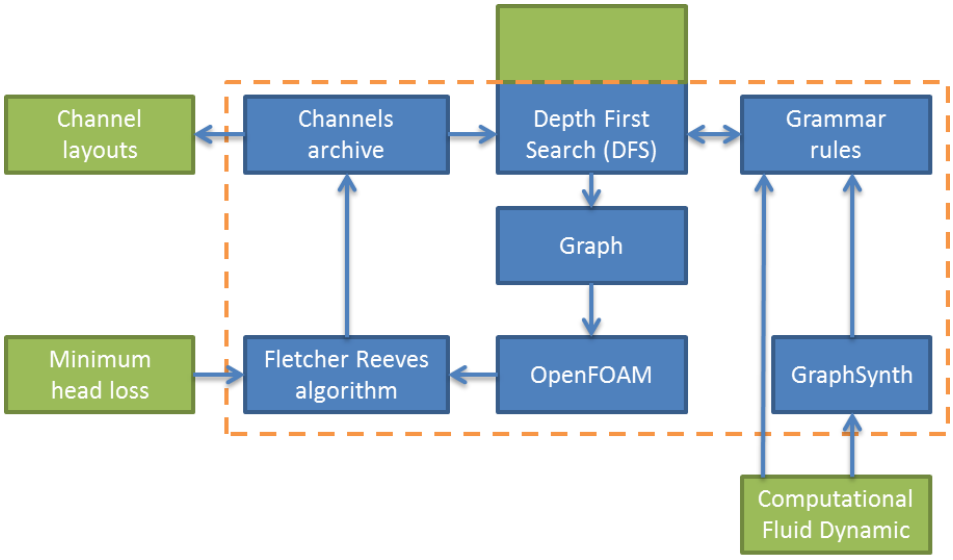

Figure 4. CDS Platform components for solving fluid channel layout synthesis

As the representation and simulation models are fully separated from each other, one can use the same rules for problems with completely different boundary conditions, fluid directions and loads. Unlike other methods, solving compressible fluids will be as easy as incompressible fluids.

Figure 5 shows two simple layout problems which have been solved by combining GraphSynth, Parasolid and OpenFOAM into the CDS Platform. The green arrows are inlets and the red arrows are outlets of the flow, which are given to the program as initial seed graphs. The program first suggests different topologies for the design; these will be filtered based on different criteria such as maximum allowed compression and only valid topologies (like the two on the left side of the Figure 5) are going to the next step. In the next step, based on the flow direction and some design parameters the shape is optimized and finally the transformer generates 3D fluid channels from representing graphs. The best solution will be found after analyzing all candidates in the CFD solver. The topic of shape and topology optimization of fluid channels with graph grammars approach has been extensively explored in a separate study by the author (Hooshmand and Campbell, 2013).

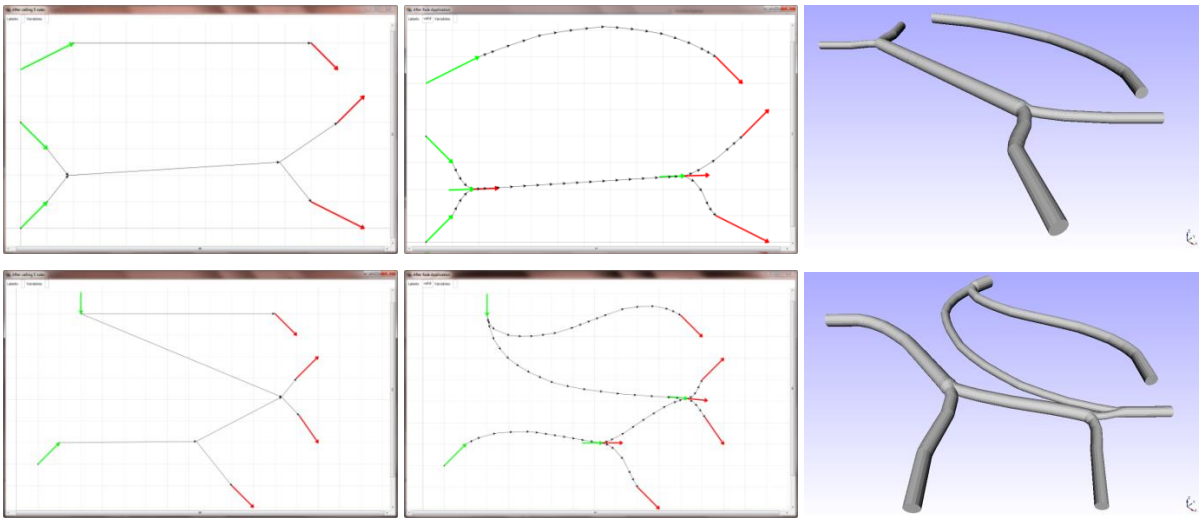

Figure 5.generating a candidate topology, and giving shape to it

There are also many other diverse areas such as synthesis of space frames and heat transfer problems, which can be explored with the platform. The next section shows results of a parametric optimization of a triangle by the platform. For solving this problem, no grammar interpreter is required since all solutions have the same topology.

4.3 Lightweight design of a triangle

Another interesting field of application that the platform can be used is parametric optimization of designs. The platform is able to cope with complicated designs with numerous parameters. The aim of this case study is to find the optimum lightweight design for a triangle with four parameters to be optimized (P1, P2, P3 and Θ). Unlike the previous two applications, a variety of recent commercial CAD packages are able to cope with this kind of problem.
In this study the Code-Aster FE solver, Salome Preprocessor and a simulated annealing algorithm have been used.

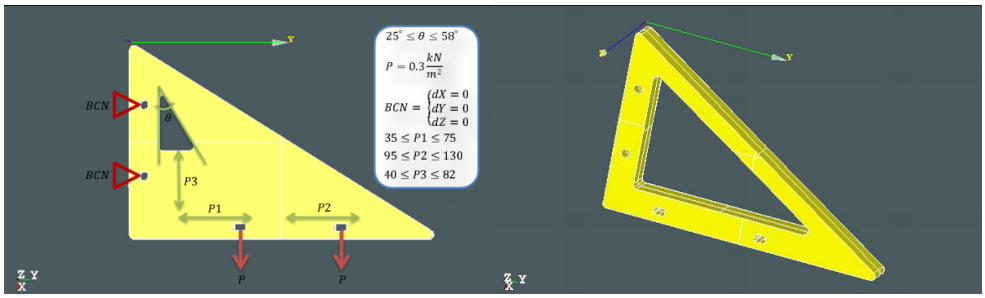

Figure 6. Forces, BCNs, and parameters, and best design after 990 iterations with Simulated Annealing (SA)

Figure 6 shows the boundary conditions and forces which are applied to the geometry and also four parameters which should be optimized (P1 to P3 and theta). The objective function for this case study is minimizing weight and stress of the triangle; objectives have different weighting factors. For meshing the geometry an automatic tetrahedralization algorithm with mesh size 5 is used. A linear statistic solver of Code-Aster is used to analyze designs after each optimization iteration. Figure 6 shows the best design after 990 iterations.

5 CONCLUSION

The Computational Design Synthesis (CDS) 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 generative design methods such as shape and graph grammars with conventional simulation and analysis methods to provide guidance in design process according to evaluated engineering criteria. The major characteristic of the presented platform that distinguishes it from all other implementations is its generality. To reach this generality and flexibility, a programming and module-based approach is used to develop the platform. The CDS Platform combines different optimization and grammatical algorithms with conventional simulation and analysis methods. The premise of this combination is to create an approach to synthesizing optimal shapes considering criteria requiring multi-physics analysis, which is required for calculating the engineering behavior of generated designs. The platform can be used in a very wide range of simulations and analyses like acoustics, finite element, computational fluid dynamic, and heat transfer and a combination of these analyses to solve complicated multi-physics problems. This has been achieved by integrating two preprocessors and solvers in the generation process; the Salome preprocessor and the Code-Aster solver for FE analysis and the snappyHexMesh preprocessor and the OpenFOAM solver for CFD and thermal analysis. Automatically integrating these analyses in the design process is a known challenge for engineers and designers. Unlike many commercial software, its object-oriented and module based implementation provide a unique possibility for designers to integrate any simulation module of the platform in their design processes in a few simple steps. The platform works like a high level API and prevents the direct interaction of designers with many complexities of the simulation and optimization packages. Its open source character gives the researchers the ability to extend, modify and customize the platform to their needs.

ACKNOWLEDGMENTS

With the support of the Technische Universität München – Institute for Advanced Study, funded by the German Excellence Initiative.

REFERENCES

Andreasen, C.S., Gersborg, A.R., and Sigmund, O. (2008) 'Topology optimization of microfluidic mixers', Int J Numer Methods Fluids vol. 61, pp. 498–513.

Bolognini, F., Seshia, A. A., and Shea, K., (2007) 'A Computational Design Synthesis Method for MEMS Using COMSOL', Proceedings, the COMSOL Users Conference, Grenoble.

Borrvall, T., and Petersson, J., (2003) 'Topology optimization of fluids in stokes flow', Int J Numer Methods Fluids vol. 41, pp. 77–107.

Cagan, J., Campbell, M.I., Finger, S., and Tomiyama, T., (2005) 'A Framework for Computational Design Synthesis: Model and Applications', Journal of Computing and Information Science in Engineering, vol. 5, pp. 171-181.

Campbell, M. I., (2006) 'The official GraphSynth web-page', http://www.graphsynth.com, University of Texas at Austin.

Code-Aster, (2012) 'The official Code-Aster web-page', www.code-aster.org.

Deshpande A., and Riehle. D., (2008) 'The Total Growth of Open Source', SAP Research, SAP Labs LLC, vol. 275, pp. 197-209.

FreeCAD, (2012) 'FreeCAD official web-page', www.sourceforge.net.

GNU operating system, (2012) 'GNU official web-page', www.gnu.org.

Helms, B., Shea, K., and Hoisl, F., (2009) 'A framework for computational design synthesis based on graph-grammars and function-behavior-structure', in IDETC/CIE 2009, August 30 - September 2, San Diego, California, USA.

Hertel, G., Niedner, S., and Herrmann S., (2003) 'Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel' Research Policy, vol. 32, pp. 1159–1177.

Hoisl, F., and Shea, K., (2011) 'An interactive, visual approach to developing and applying parametric three-dimensional spatial grammars', Artificial Intelligence for Engineering Design, Analysis and Manufacturing, vol. 25, pp. 333–356.

Hooshmand, A., Campbell, M.I., and Shea, K. (2012) 'Steps in Transforming Shapes Generated With Generative Design Into Simulation Models', IDETC/DAC 2012, August 12-15, 2012, Chicago, Illinois, USA.

Hooshmand, A., and Campbell, M.I., (2013) 'Topology Optimization of Fluid Channels Using Generative Graph Grammars', IDETC/DAC 2013, August 04-07, 2013, Portland, OR, USA.

Karac, A., (2003) 'Drop impact of fluid-filled polyethylene containers", Imperial College London, PhD. Thesis, June 2003.

Khanjani, A., and Sulaiman, R. (2011a) 'The aspects of choosing open source versus closed source', Computers & Informatics (ISCI), 2011 IEEE Symposium, pp. 646-649, March 20-23.

Khanjani, A., and Sulaiman, R., (2011b) 'The process of quality assurance under open source software development', IEEE Symposium on Computers and Informatics.

Kurtoglu, T., Swantner, A., and Campbell, M.I., (2010) 'Automating the conceptual design process: From black box to component selection', Artificial Intelligence for Engineering Design, Analysis and Manufacturing, vol. 24, special issue 01, pp. 49-62.

Liu, Z., Gao, Q., Zhang P., Xuan M., and Wu, Y., (2011) 'Topology optimization of fluid channels with flow rate equality constraints', Struct Multidisc Optim, vol. 44, pp. 31–37.

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.

MITRE Corp., (2003) 'Use of Free and Open-Source Software (FOSS) in the US Department of Defense' Report by MITRE Corp, 2003.

OpenFoam, (2012) 'OpenFOAM official web-page'', www.openfoam.com.

Rai, R., Kilaru, P., Vallepalli, R., and Campbell, M. I. (2011) 'A Novel Search Algorithm for Interactive Automated Conceptual Design Generator (ACDG)' ASME, vol. 5: 37th Design Automation Conference, Parts A and B, Washington, DC, USA, August 28–31, 2011.

Ribes, A., and Caremoli, C., (2007) ‘Salome platform component model for numerical simulation’, EDF R&D, Computer Software and Applications Conference, 2007.
Ruffin, M., and Ebert, C., (2004) ‘Using Open Source Software in Product Development: A Primer’, IEEE Software, vol. 21, issue 1, pp. 82-86.
Salome, (2012) ‘Salome Platform tutorial’, www.salome-platform.org.
SciPy, (2012) ‘SciPay official web-page’, www.scipy.org.
Shea, K., and 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.
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.
Starling, A. C., and Shea, K., (2005) ‘A Parallel Grammar for Simulation-Driven Mechanical Design Synthesis’, Proceedings, ASME Long Beach, California, USA, September 24-28, 2005.
Vangelooven, J., Malsche, W.D., Beeck, J.O.D., Eghbali, H., Gardeniers, H., and Desmet, G., (2010) ‘Design and evaluation of flow distributors for microfabricated pillar array columns’, Lab Chip, vol. 10, pp. 349–356.
Wang, X.J., Zhang, C.Y., Gao, L., and Li, P.G., (2008) ‘A Survey and Future Trend of Study on Multi-Objective Scheduling’, IEEE, ICNC '08. Fourth International Conference on Natural Computation, vol. 6, pp. 382–391.