Scripting Design Supported by Feedback Loop from Structural Analysis

Analytical model

Lukáš Kurilla[1], Henri Achten[2], Miloš Florián[3]
Faculty of Architecture, Czech Technical University in Prague, Czech Republic
[1]http://kurilluk.net, [2]http://www.molab.eu, [3]http://www.studioflorian.com
[1]mail@kurilluk.net, [2]achten@fa.cvut.cz, [3]milos.florian@fa.cvut.cz

Abstract. *In order to support an architect's decision to evaluate and choose more efficient structural solutions in the concept design, it is necessary to establish an interactive feedback loop between structural solver and geometry modeller which would allow one to analyse a great number of solutions generated in the scripting design process. Defining a cross-disciplinary data structure as an analytical model, the communication between existing structural solver (OOFEM) and geometry modeller (Grasshopper) was established. Automation of the entire analysis process was done by the bridging tools MIDAS and Donkey, which have been developed. This paper presents the method of creation of an analytical model by Donkey, and deals with how to visualize, interpret and use the result values from the structural analysis.*
Keywords. *design tool development; computing design; decision-making support methods; finite element method; cross-disciplinary cooperation.*

INTRODUCTION

"Scripting Cultures considers the implications of lower-level computer programming (scripting) as it becomes more widely taken up and more confidently embedded into the 'design process' ... scripting affords a significantly deeper engagement between the computer and user by automating routine aspects and repetitive activities, thus facilitating a far greater range of potential outcomes for the same investment in time" (Burry 2012).

By exploring a wider range of design alternatives, one can find more efficient structural solutions directly in the concept design phase, reducing the costs of other phases, including production costs. It is therefore important to support architects' decision-making with structural analysis so that, through creating a wide range of solutions, they can find as close to an optimal solution as possible.

In comparison with state-of-the-art automated or interactive optimization processes (Buelow, 2009; Shea, 2005), this approach generates solutions directly by the designer with full design control. Although this process will generate fewer solutions than the optimization process, the solutions will, however, reflect more of the architect´s preferences, individuality, and feelings (Figure 1).

To support architects' decision-making in the scripting design process, we need to find answers to these research questions:

- How to establish analysis feedback loop and get results directly to architects?
- How to correctly interpret structural analysis results without requiring sufficient professional knowledge?
- How to support decision-making in the design process using analysis results?

FEEDBACK LOOP

To create a feedback loop that brings structural analysis results directly to architects, we need to automatize the whole analysis process. This is done by bridging an existing modeller and solver with the tool we have developed (Figure 2a). To establish communication between the bridged tools we define cross-disciplinary data structure (analytical model, Figure 2b).

For cross-disciplinary data structure, the IFC (Industry Foundation Classes) standard might be also used (Eastman, et al., 2011), but like some other workflow approaches (Af Klercker and Pittioni, 2002) it is more suitable for the final design phase than in the concept design, because architects create more complex geometric models and expect final assessment from structural engineers. Our approach is more suitable with specific model for specific analysis in different design phases (Svoboda, 2013).

Bridging existing software tools

Visual programming environments with CAD packages can be very effective for shape exploration through real time generation of parametric variations (Celani and Vaz, 2012). Therefore, to create the analytical model, Grasshopper (GH) has been chosen which is integrated with the NURBS-based 3D modelling tool Rhinoceros. GH is a visual programming tool popular among academics and professionals. It allows designers to generate complex geometries, still preserving the possibility of interactive modifications. Donkey is a plugin of GH developed to create analytical model data in the defined data structure, read analysis results and visualize them to support architects' decisions in the design process. It is written in C# and uses Rhino and GH SDK libraries [1]. To analyse the analytical model, Donkey communicates with MIDAS.

MIDAS has been developed on the structural engineering side by Svoboda L. It is a tool without graphic user interface designated for manipulating both input and output models of structural analysis. It is written in C++ and released under GPLv3 license [2]. MIDAS as a converter allows the conversion of different input and output models and thus ensures modularity of the bridging tool (Svoboda, 2013; Kurilla, 2012) (Figure 3). The solver and modeller pl-

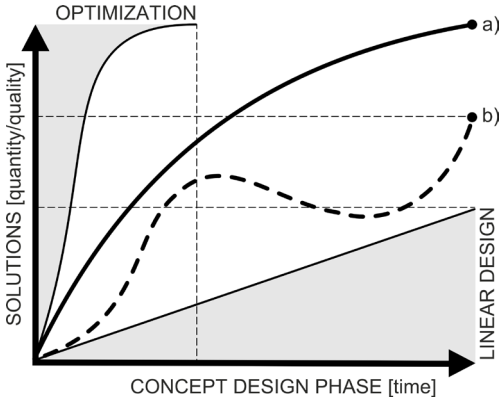


Figure 1
Comparison of design processes, Scripting design: a) with decision-making support, b) without decision-support.

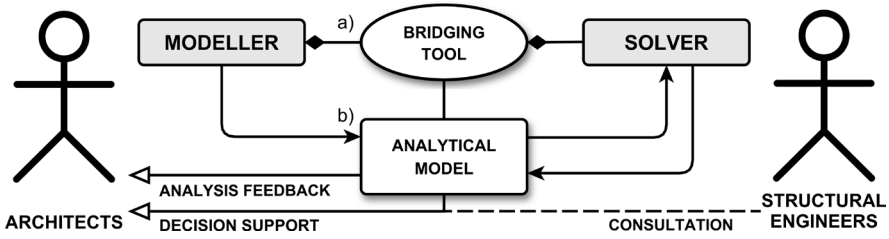
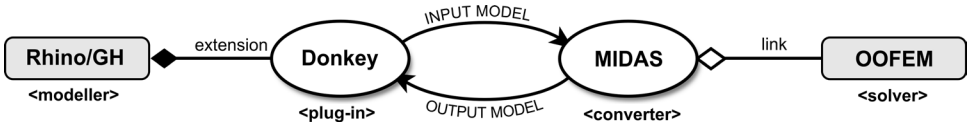


Figure 2
Feedback loop, proposed workflow: a) bridging existing tools with our developed tool, b) defining cross-disciplinary data structure as an analytical model.

Figure 3
Bridging software tool architecture, modular approach.



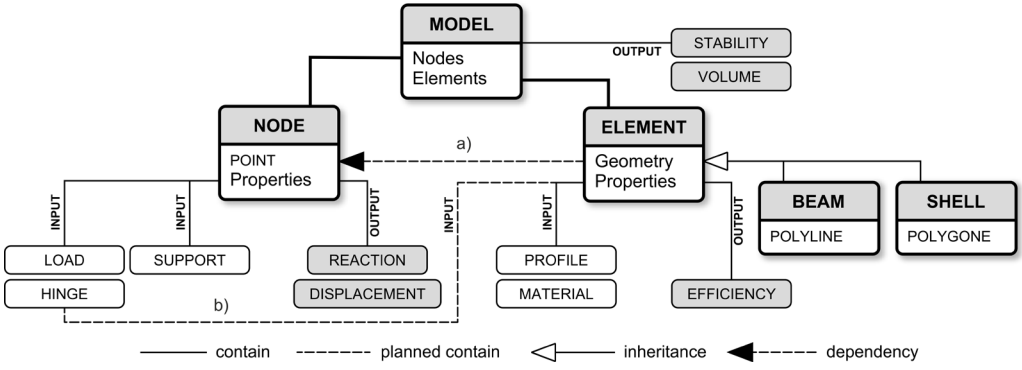
ugin in this case are modules of the modular system, which allows MIDAS to calculate analysis in different solvers or get models from different modellers. As a main solver for structural analysis, we use OOFEM, which is directly linked by MIDAS as a dynamic library. OOFEM is a modular finite element code for solving problems of solid, transport and fluid mechanics. It is released as the open source software operating on various platforms [3].

Analytical model data structure

For the analytical model, an object oriented approach was chosen (Figure 4). A MODEL is the root object of the data structure. It contains a list of two basic objects: NODES and ELEMENTS of the structure. Each of these objects has its own specific properties and geometric representation. Geometric representation of the nodes is a point. The geometry of elements depends on the points, which ensures a clear definition of connections between elements (see below in the model definition, component model) and allows direct response of an element's shape to the change in the nodes position. Geometric representation of elements varies according to the type of element (beam, shell). From the real shape of an element the smallest (proportionally negligible) dimension is excluded and in the form of PROFILE is stored in the element properties. A beam is thus represented as a line (1D element) and a shell as a surface, polygon (2D element). Geometric simplification has the advantage especially in reducing time-consuming calculations, and also makes possible rapid changes in the size of an element.

In the element properties, in addition to the profile, as input data, information about the material is also stored. This input data is analysed and the results are returned to the model in the form of output data. Thus, the resulting efficiency value of the profile and material usage/utilization is added to the properties of the element (see interpretation of the analysis results). For the node, input data is represented by boundary conditions of the structure, which include: support, loads, and hinges. Depending on these, after analysis, responses and displacement of the structure are stored as output properties of nodes. In the future work we plan to define some boundary conditions also on the element, which will bring more flexibility to model definition (Figure 4b, see analytical model definition). For the entire model, in the analysis process, the volume of

Figure 4
Analytical model data structure, object oriented approach, a) structural elements can share common nodes (geometric dependency) b) boundary conditions (hinges, loads) can be defined on elements (planned contain).



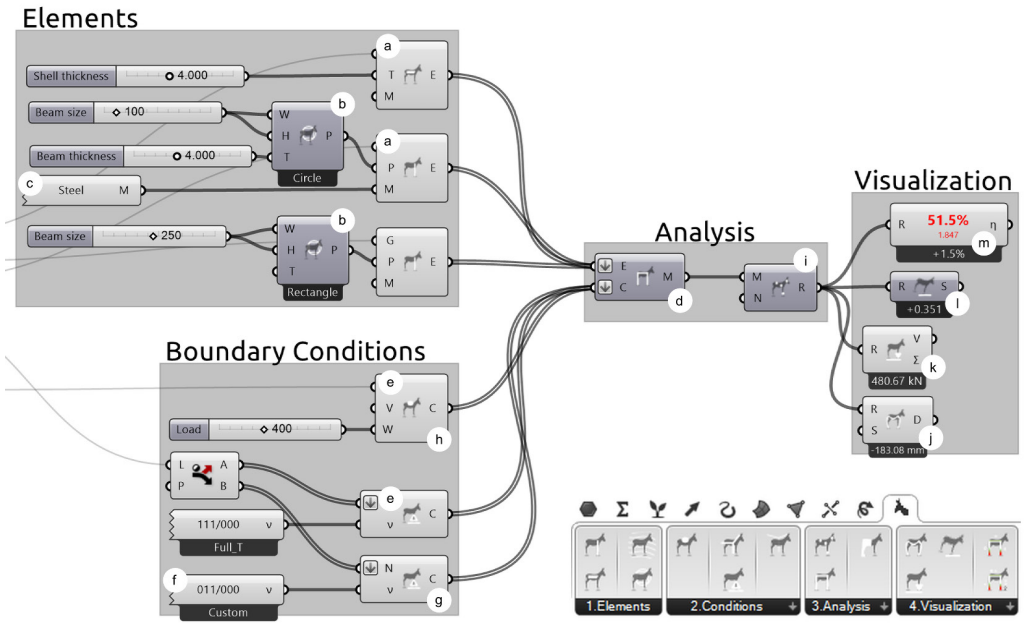


Figure 5
Donkey components, script definition: a) element geometry definition, b) profile, c) material, d) model, e) node/element boundary conditions, f) degree of freedom, g) support, h) load, i) analysis, j) displacement, k) reactions, l) stability, m) efficiency.

the structure (material consumption) is calculated and the stability of the whole structure is assessed. Stability is currently calculated only for a structure consisting of beam elements. The used OOFEM (solver) does not yet support shell buckling (soon to be completed, hopefully). However, due to the modularity of the system, it is possible to calculate the shell buckling in ANSYS.

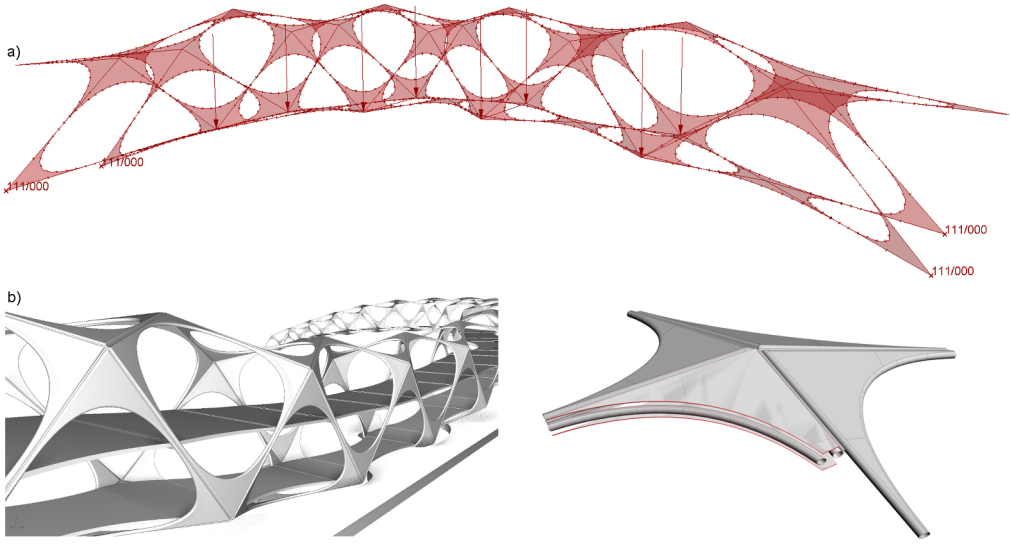
DONKEY

As mentioned in the previous sections, Donkey is a plugin to GH, which provides a structural analysis feedback loop. Compared with similar plugins in GH, like Karamba and Millipede [4, 5], it is an open source program supporting architects' decisions in the design process. It focuses mainly on the interpretation of the results of structural analysis, so they can be correctly interpreted without requiring extensive professional knowledge. The structure and interconnection of the developed components varied throughout the project process. When designing with respect for logic and functionality, GH was especially taken into account (not to change the habits of users) but mainly to create a script definition for analysis that is easy to use, in other words, not to discourage a user, giving a poor first impression. Donkey components are arranged into four groups of GH's menu: elements, conditions, analysis and visualization (Figure 5).

Analytical model definition

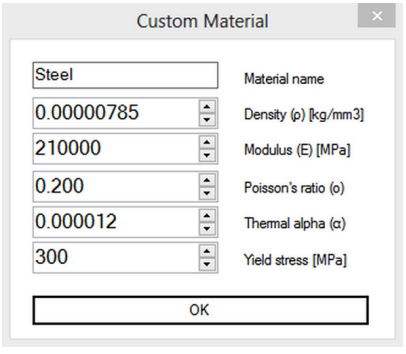
The first group of components is for the structural elements definition. We can choose from two mentioned types of elements (shell and beam) and link elements with geometry created directly in GH or in the Rhino (Figure 5a and 6). In the beam element properties, it is possible to define two basic shapes of the PROFILE (square, rectangle, Figure 5b), set their dimensions (width, height) and define the thickness to have a pipe profile. For shell elements the thickness value is only needed. Donkey does not support more complex shapes of profiles such

Figure 6
Analytical model geometry representation a) analytical model visualization b) architectural model.



as T, I. They need more complex calculations to get more precise results. We assume that it is not necessary to work with more complex shapes of profiles in the concept design, where the shape, topology and preliminary dimensions of the structure are being explored. On the contrary, the quick feedback is more important in this phase, so we do not plan to add more complex shapes to the tool for now. You can choose the MATERIAL (Figure 5c) from basic predefined materials or you can create your custom material by defining density, modulus, Poisson's ratio, thermal alpha and yield stress values (Figure 7). Similarly to the profile, though it is possible to define custom material, due to following post-process calculation to simplify analytical result interpretation, the result may not be precise (see the interpretation of the results).

Figure 7
Custom material setup, pop-up window.



The MODEL component merges the defined elements into one complex structure (Figure 5d). Points of each element are registered as structural nodes in the model. If the node already exists, the point of the element merges with it (merging depends on the tolerance setup in Rhino). Thus the geometry dualities are removed (common mistake in modelling) and clearly defined connection between elements is secured (elements share common node). When the elements are loaded, the MODEL component informs about the missing definition of boundary conditions of the structure, precisely about support, which is necessary for the analysis calculation (an unsupported model cannot be calculated). The boundary conditions are defined by their specific properties and relation to the node or to the structural element (Figure 5e).

The degrees of freedom (DOF) is a specific prop-

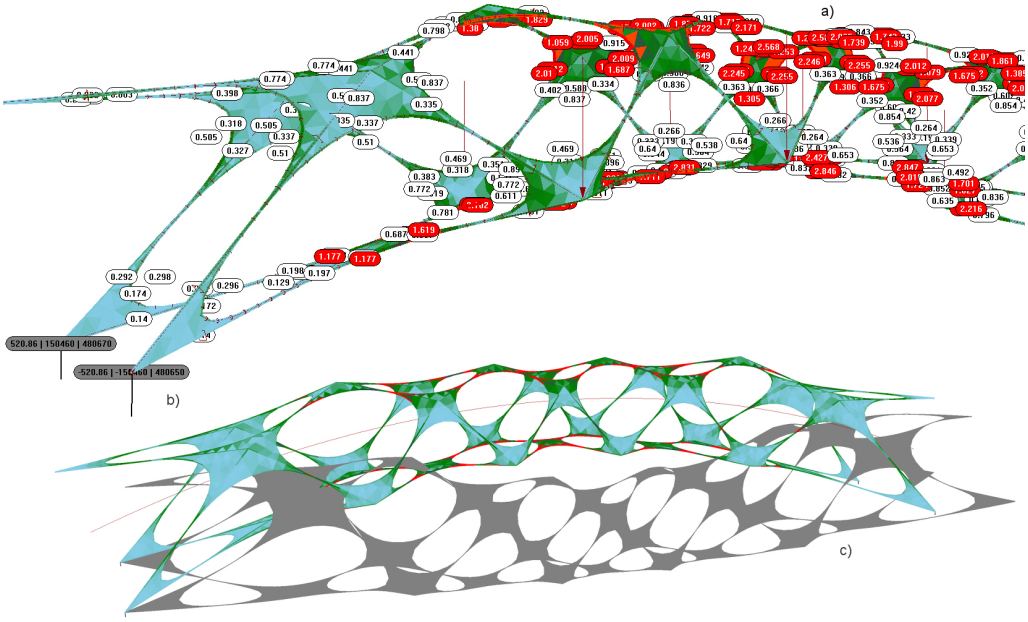


Figure 8
Analytical model resulting data visualization: a) efficiency, b) reactions c) displacement.

erty concerning SUPPORTS and HINGES. Like the material, the DOF can be defined by choosing preset frequently used options or by creating a custom definition. Custom DOF is defined by six check buttons, where check removes degree of freedom of movement or rotation in the selected axis. Currently we use only full DOF for the hinge. In our future work we plan (providing that hinge – element link exists, Figure 4b) to define DOF for a hinge and determine its relation to the adjacent elements. This is necessary for a definition of a hinge that connects more than two elements.

The analysis automatically deals with the deadweight of the elements. Additional load can be added by NODAL LOAD. It is necessary to specify two properties: a vector determining the direction of the force and weight in kg determining load quantity (weight is automatically recalculated to the Newton). In our future work we plan to add AREA LOAD for that. Similarly to the hinge, it is necessary to ensure the link with the element (Figure 4b).

Visualization and interpretation of the analysis results

The created model can be analysed using the ANALYSE component (Figure 5i), calling MIDAS on the background, and retroactively loading the results of the analysis. These results can be visualized using the visualization component group. DISPLACEMENT is displayed as the shift of the structure (Figure 5j and 8a), which can be increased and reduced using the scale parameter (scale 1 corresponds to actual deformation of the structure). The output value of displacement component is the maximum deformation of the element or the entire structure in mm. REACTIONS represent the size and direction of force acting at the foundations of the structure (Figure 5k, 8b). They are displayed using three direction vectors at the supports. The output value is the series of vectors (one for each support), with size corresponding to the force in Newton. STABILITY (Figure 5l), the resulting value is a dimensionless coefficient representing a multiple of the actual load at which the

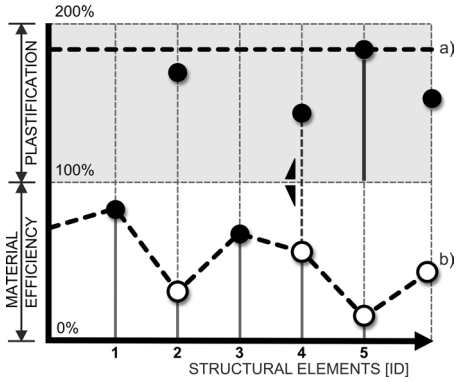


Figure 9
Result values for efficiency component: a) penalty value (MAX - 1), b) rate value (AVERAGE) %.

particular structure loses stability. If the coefficient is within the interval 0 to 1 the structure is unstable (a lower number means greater loss of stability). If the coefficient is negative (< 0), the structure is not at risk (at all) of losing stability (mostly simple structures).

EFFICIENCY value represents the percentage of used profile/material of the element (Figure 5m). It is used to facilitate the correct interpretation of the FE analysis results (including moment and normal forces), where insufficient knowledge and experience result in the incorrect interpretation of results (focusing on normal forces only, etc.).

The efficiency value is calculated in post-calculation in MIDAS, and based on the von Mises yield criterion where the element's stress components are integrated into one equivalent stress and divided by the yield stress of a defined material. With this calculation method we are very precise for steel materials, and we are fully aware of this value being less precise, especially for materials of anisotropic strength. However, the aim of the analysis is not to get a final, precise assessment, but only to guide architects in their decisions in the design process. In this respect, the analysis provides us with instant and sufficient information about the overall stress distribution in the entire model, which helps architects realise problematic parts of their design. In the future, we will attempt to find and implement in the calculation the approximation methods that will not affect the calculation demands, but will improve the accuracy of the result for basic specific materials.

The resulting efficiency value is at the interval from 0.0 to infinity. When the value is greater than 1.0 (100% efficiency), the marked element is in inadmissible plastic zones and is irreversibly deformed or broken. In the process of analysis, the geometry of the element is split into smaller segments (Svoboda, 2013). Each of the segments of the element is assigned an efficiency value, indicated by colour in the visualization of element geometry. The colour scale is based on gradient, where 0 corresponds to light blue – inefficiently used element. Green colour corresponds to 1, i.e. efficiently (100%) used/designed element. Values exceeding 1 are red, meaning problematic elasticity of the element. For each element a single value of efficiency is assigned. This is the maximum value achieved in individual element's segments. This can be displayed as text corresponding to element geometry (Figure 8a).

When displaying value for a component, the question might arise: how to calculate the resulting efficiency for the entire model? In the current solution, a penalty function was selected for the calculation. The result is thus split into two values (Figure 9). The first represents the average of the element values within the interval of 0 to 1 (up to 100%), whereas values higher than 100% are correspondingly converted (equivalent distance from 1) into the above interval. The second value of the result represents the penalisation of the previous assessment. Its value represents maximum elements of the model in the plasticisation zone (above 100%).

Decision-making support

Two methods that support decision-making in the design process, are currently implemented in Donkey. The first method compares a current solution with the last one (sensitive evaluation), the second method compares it with the best solution created in the design process so far. The architect gets information about the result of comparison in the message box, placed at the bottom of the respective component.

For example, the displacement component provides information about the difference in values of maximum model deformation, and informs the architect about improvement or deterioration of the structure deflection considering the last updates.
To evaluate the current solution with the best solution a specific EVALUATE component was designed (Figure 10). The component records and evaluates all solutions throughout the design process. If the current solution is less suitable, the component will show the best solution to guide/inspire architects (Figure 11). The results of comparison are also shown in the message box to support sensitive changes.

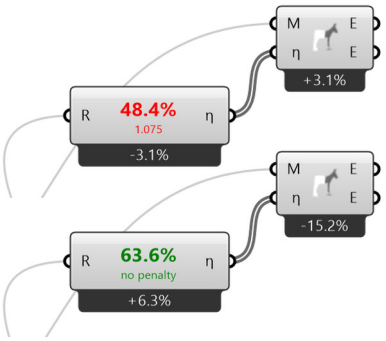


Figure 10
Evaluation component evaluates the actual model with the best found solution in efficiency criteria.

FUTURE WORK AND CONCLUSION

When comparing the current solution with the best solution, ranging from efficiency to the overall comparison of models (considering all properties) we enter a new broad topic of multi-criteria evaluation. Implementation of this evaluation method as well as implementation of improvements mentioned in the previous sections of this paper into the developed tool, is our future aim and work.

We have established the system, which brings structural analysis feedback to architects' modeller in understandable form to easily and correctly interpret the analysis results. It also teaches architects to better understand the behaviour of the structure they designed and helps them improve it. This makes it possible to create more efficient solutions directly in the concept design phase.

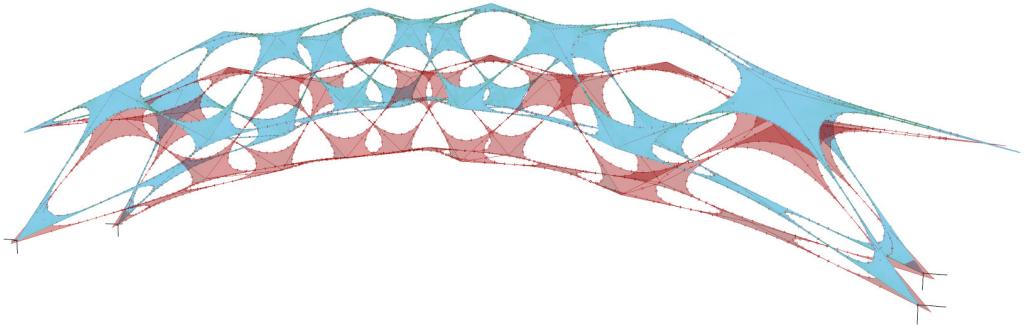


Figure 11
Preview of evaluation the actual model with the best found solution.

ACKNOWLEDGEMENT

The authors thank Ladislav Svoboda for cooperation and developing MIDAS module. We also gratefully acknowledge the endowment of the ministry of industry and trade of the Czech Republic under project FR-TI1/568. We would like to extend special thanks to Reinhard König from Chair of Information Architecture ETH in Zurich, for rewarding and fruitful discussions. Finally we would like to thank Matěj Lepš, Jan Zeman and Jan Novák from Faculty of Civil Engineering CTU in Prague for their consultations and collaboration.

REFERENCES

Af Klercker, J. and Pittioni, G. (2002), 'Architect and Structural Engineer in interactive design', in *Connecting the*

Real and the Virtual - design e-ducation [20th eCAADe Conference Proceedings], Warsaw, Poland, pp. 386-389.

von Buelow, P. 2009, 'A comparison of methods for using genetic algorithms to guide parametric associative design', *In Proceedings of the International Association for Shell and Spatial Structures (IASS) Symposium 2009*, Valencia, pp. 11.

Burry, M., (ed) (2011), *Scripting Cultures: Architectural Design and Programming*, John Wiley & Sons Ltd, West Sussex, pp. 8-9.

Celani, G. and Vaz, Carlos Eduardo 2012, 'CAD Scripting and Visual Programming Languages for Implementing Computational Design Concepts: A Comparison from a Pedagogical Point of View', *International Journal of Architectural Computing*, Journal of Architectural Computing, Vol. 10 Issue 1, p. 128.

Eastman, C.; Teicholz, P.; Sacks, R. and Liston, K. (2011), *BIM handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors*, 2th. Edition, John Wiley & Sons, Inc., New Jersey.

Kurilla, L., Ruzicka, M. and Florián, M. 2012 'Architectural Software Tool for Structural Analysis (ATSA)', Digital physicality - *30th eCAADe Conference Proceedings*, Prague, Czech Republic, pp. 547-553.

Svoboda, L., Novak, J., Kurilla, L. and Zeman, J. in press, 'A framework for integrated design of algorithmic architectural forms', *Advances in Engineering Software*. [http://dx.doi.org/10.1016/j.advengsoft.2013.05.006]

Shea, K,, Aish, R. and Gourtovaia, M. 2005, 'Towards integrated performance-driven generative design tools', *Automation in Construction*, 14(2), pp. 253-264.

[1] Kurilla, L., Donkey, http://donkey.kurilluk.net/

[2] Svoboda. L., MIDAS, http://midas.igend.cz/en

[3] Patzák. B., OOFEM, http://www.oofem.org/

[4] Karamba3D, http://www.karamba3d.com/

[5] Millipede, http://www.grasshopper3d.com/group/millipede