Proceedings of the International Association for
Shell and Spatial Structures (IASS) Symposium 2013
„BEYOND THE LIMITS OF MAN"
23-27 September, Wroclaw University of Technology, Poland
J.B. Obrębski and R. Tarczewski (eds.)

Open source engineering and sustainability tools for the built environment

Jeroen L. Coenders[1,2]

[1] Founding Leader, BEMNext Lab, Faculty of Civil Engineering and Geosciences, Delft University of Technology, Delft, The Netherlands, *jeroencoenders@gmail.com*
[2] Associate, Computation Leader, Arup, Amsterdam, The Netherlands, *jeroen.coenders@arup.com*

Summary: This paper presents two novel open source software developments for design and engineering in the built environment. The first development, called "sustainability-open" [1], aims on providing open source design, analysis and assessment software source code for (environmental) performance of buildings, structures and infrastructure. The aim of this project is to take away barriers that might exist to design and engineer buildings, structures and infrastructure in a sustainable manner and based on quantitative measures or metrics of (environmental) performance for a variety of design aspects. The second development, called "engineering-open" [2], aims on providing open source software source code for design, calculation, form finding, analysis and optimisation of structures. Both developments aim to take away restrictions that might exist for science and practice caused by the unavailability of open and insightful software source code to inspire further research, development and innovation in the architecture, engineering and construction industry for new software approaches and tools.

Keywords: *digital design, design tools, engineering, sustainability, open source software, parametric software, BIM, finite element method, form finding, optimisation*

1. INTRODUCTION

This paper presents two novel open source software initiatives that aim to take away restrictions that might exist for science and practice caused by the unavailability of open and insightful software code for the built environment.

The first initiative, called "sustainability-open" [1], aims on providing open source design, analysis and assessment software source code for (environmental) performance of buildings, structures and infrastructure.

The second initiative, called "engineering-open" [2], aims on providing open source software source code for design, calculation, form finding, analysis and optimisation of structures.

Both developments have been initiated by the BEMNext Lab at Delft University of Technology which aims to contribute to the next generation of software for modeling of the built environment.

It must be noted that both initiatives are constantly being developed and that therefore at time of publication the source code repositories might be developed further than this paper currently describes. Please refer to the software code repositories for the latest version of the initiatives.

1.1. Open source

The software industry as well as the academic world has produced many applications that designers and engineers can use to design, analyze, engineer, manage and control their designs for the built environment. However, often the source code of these applications is not available to the public due to the closed-source and propriety nature of the application. The disadvantage of the closed-source approach is that the designers lack insight in the internal workings of the software application rendering first hand validation of the algorithms and techniques applied in the application virtually impossible. Only black-box validation of results can be performed to gain trust in the correct workings of the software. Exactly this lack of trust has been identified as one of the major barriers for adoption of more and more advanced software in design and engineering [3] with the result that the building industry is missing out on many advantages to be gained by a wider-spread adoption of information technology in the design, engineering and construction process. For the academic world where validation of results is important to achieve a high grade of scientific quality and where reproducibility of results is an important measure of quality, the closed-source nature of these software applications forms a similar blockade. In contrast to the closed-source approach provides an open source approach to software code full insight in the methods used. Designers are able to access the methods used by directly inspecting and reviewing the programmed logic rather than depending only on black-box results. Academics are able to reproduce results because of the availability of the software with the additional advantage that the research might be extended by further research without re-inventing (part of) the wheel.

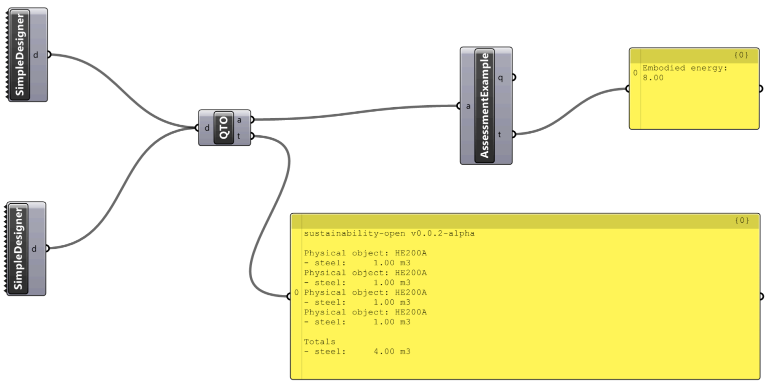
Another advantage of an open-source approach is that the shared knowledge can act as a base for learning and can form a source of inspiration for further research, development and innovation which builds on the available source code and encoded knowledge.

2. SUSTAINABILITY-OPEN

The first software development aims to provide open source software code so that the wheel does not have to re-invented, the methods are open for review and validation, and reproduction of results. Furthermore, although strictly speaking not an advantage of open source software, this development provides the software code at no cost, taking away the barrier of cost of software acquisition or development. "sustainability-open" (http://www.sustainability-open.com) is a research project and software development initiative [1] that aims on providing open source software code that can be used to build software applications for the built environment to produce sustainable designs by application of digital design methodologies.
The aim of the initiative is to take away the reason design and engineers might have of unavailability of knowledge and software (tools), which might prevent them from designing sustainable buildings, structures and infrastructure. By providing the software code for environmental performance measurement (and other aspects of design and engineering) and related design and analysis technology the threshold to produce sustainable designs becomes very low.

This initiative deliberately does not take a stance on how to measure sustainability, but believes that opening up the metrics will give decision makers the insight to make the best choices. The initiative provides a base framework for everybody to develop his or her own measures, such as material use (measured in quantity or embodied energy), material depletion, energy usage (embodied, operating, transport, life-cycle), pollution, radiation, toxicity, emissions, etc. The framework does not only focus on the 'negative measures' that measure environmental performance in terms of an 'environmental cost', but can also include measures to measure 'environment gains' or 'benefits', such as recycling, up-cycling, local energy production and storage, and soft measures, such as adaptability and comfort (a more adaptable or comfortable building will be in use longer than a less adaptable or comfortable building).

The development consists of a number of software layers. A distinction is being made between the core framework, the connection to user interface (or the application that provides this connection) and add-ons.



```
Embodied energy:
8.00
```

```
sustainability-open v0.0.2-alpha

Physical object: HE200A
- steel:     1.00 m3
Physical object: HE200A
- steel:     1.00 m3
Physical object: HE200A
- steel:     1.00 m3
Physical object: HE200A
- steel:     1.00 m3

Totals
- steel:     4.00 m3
```

Fig 1. Grasshopper model showing the current three layers of the framework: Designers, Analysis (QTO component in this case) and Assessment components. The yellow boxes are Grasshopper [6] textual output components.

2.1. Framework design

The core framework contains the base functionality to perform the design, analysis and assessment actions and is independent from both the interface and domain logic that exists in add-ons.

This means that the framework does not contain much functionality itself, but provides the scaffolding for others to build on and it drives the process of calling the different actions that the components perform in the right order.

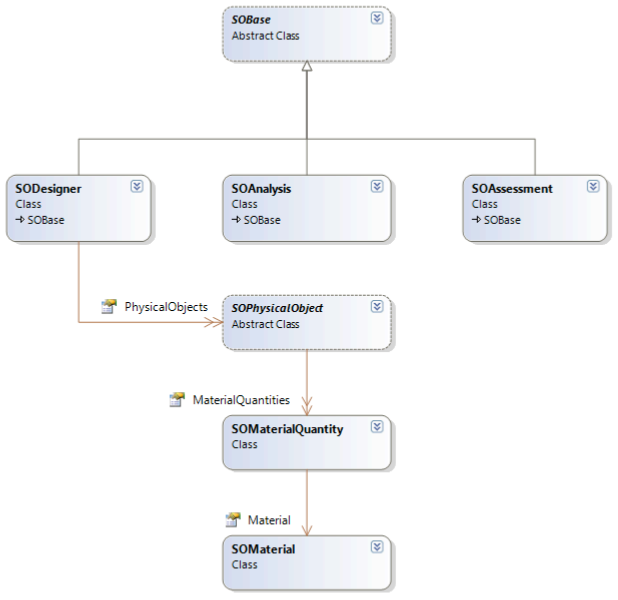


Fig 2. UML class diagram of the sustainability-open framework

The current development contains three areas of technology that can be combined in different design and engineering approaches with different user experience, awareness of performance and ability to influence the design outcome, during the design process:

Automated design technology (also: designers or designer components) provides components and methods to let the application automatically produce a (partial) design of (a part of) the total design. The designers are derived from the SODesigner class (see also Fig 2 and 3). On these designers the RunDesigner method is called by the framework, which contains the actions to build the design.

A design is stored as a set of 'physical objects' (SOPhysicalObject class), which contains material quantities, which specifies the amount of a certain material (see also Fig 2).

Analysis methods (also: analysis components) provide components, algorithms and methods to (connect to) analysis methods that produce results based on the design input or other analysis results.

An example of an analysis method is for instance a quantity take off analysis (QTO), which produces a bill of quantities for the total design. This analysis method is one of the few implemented components that is packaged with the framework as it is required by many other components.

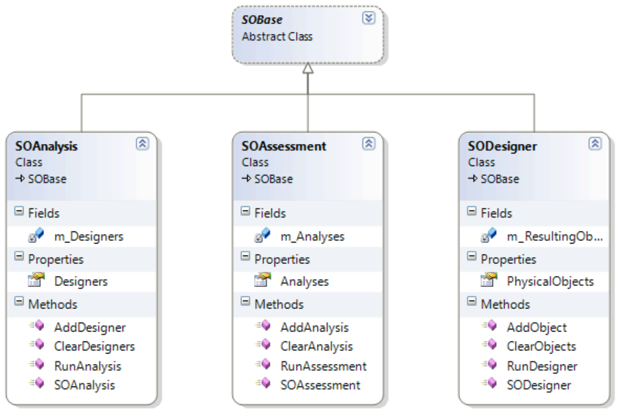


Fig 3. Detailed UML view of the three main classes in the sustainability-open framework: SODesigner, SOAnalysis and SOAssessment.

Assessment methods (also: assessment components) provide components, algorithms and methods to assess the design based on analysis results or other assessments.

An example of an assessment method is an embodied energy assessment, which takes the bill of quantities from the QTO analysis and calculates the total amount of embodied energy in the design.

Note that the framework and its components described above with the exception of the QTO analysis do not perform any actions. These actions will need to be implemented in components part of add-ons. These components will contain the encoded knowledge and can be easily exchanged to get maximum flexibility in the method of assessing the environmental performance of a design.

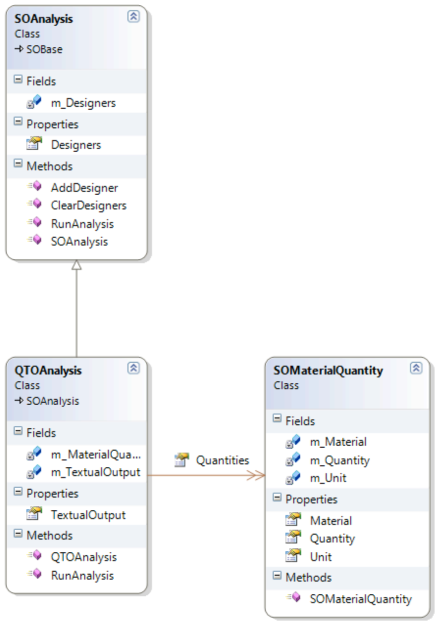


Fig 4. Detailed UML view of the Quantity Take Off (QTO) analysis

2.2. Add-ons (also called: Components)

The research project also provides a variety of implemented automated design, analysis and assessment components for (automated) structural design and assessment of steel office buildings with a rectangular floor plan and measurement of embodied energy in the form of open source add-ons. These components are called the 'BEMNext Lab components' because they are part of the original research project.

The approach however is not limited to these add-ons. Any person or company can join the community by building his or her own add-ons with specific domain knowledge and sharing these with the rest of the world.

2.3. Implementation

The current core framework has been developed in C# and the .NET framework [4]. It is open source and can be distributed with little restriction. The current connection to a user interface is based on the parametric and associative application Grasshopper [6], which is used to provide an easy-to-use and accessible user interface to build custom models.

It needs to be noted that this paper describes version 0.0.1 of the framework implementation. The source code of the framework and the components created by the BEMNext Lab are hosted on Github [5].

2.4. Licensing

The sustainability-open initiative is released under different open-source licenses. The framework is released under the Apache 2.0 license, which gives the user much freedom to use the source code in their own projects, either commercial or non-commercial, without the need of sharing the source code of these developments. For each of the add-ons different compatible licenses can be chosen. The BEMNext Lab components will be released under a license that requires copy-left as we try to keep our knowledge open for scientific and innovation purposes.

2.5. Future developments

Open source projects are under continuous development, so while the paper describes version 0.0.1 of the framework, many plans exist for expanding and extending the framework as well as various new add-ons and components. Below a number of these plans will be discussed:

- The dependency between the core framework and Grasshopper will be removed so that the framework can be used independently from Grasshopper. This is important to make the framework work with other software applications. It needs to be noted that although the dependencies will be removed, the framework will not stop working with Grasshopper.
- The above-mentioned development will make integration with other parametric software applications possible, but more importantly integration with Building Information Modeling (BIM) software [7]. As BIM adoption is rapidly spreading through the industry, it is important for the framework to work with this category of software. This will make it easy for designers to model their design in BIM software and analyze and assess with sustainability-open.
- As BIM software often depends heavily on manual modeling rather than automation of design, manual import of design objects into the framework will be made possible. This feature will also be useful for static parts of the design, such as the context or environment.
- A building component library will be added to the framework so that components can be pre-developed with all knowledge encoded and easily shared. This will also help the connection to BIM software, which also includes component libraries.
- A 'design' object will be added to the framework. The current framework spreads the objects in the design over the different designer components making complicated enumeration over the designers to retrieve all physical objects in order to perform analysis or assessment. A design object will simplify implementation of future add-ons and components, which perform analysis or assessment on the design.
- Current implementation is restricted to three-layer models consisting of designers, analysis components and assessment components. Multiple or missing layers are not allowed. In the future the implementation will be extended with designer-to-designer, designer-to-assessment, analysis-to-analysis and assessment-to-assessment relationships so that more complicated model arrangements can be build.

3. ENGINEERING-OPEN

The second software development is called "engineering-open" [2] and aims to provide open source software code for typical design and engineering algorithms, such form finding, (structural) analysis and optimisation, for application in buildings, structures and infrastructure so that designers and engineers in practice and academia can easily build and extend applications to design structures that require this technology. The initiative aims to make knowledge for the design and engineering of shell and spatial structures more accessible, make the algorithms available for review, validation and benchmarking, provide the software to research and education and sparking innovation in this field by sharing the current state-of-the-art (the principle of standing on the shoulders of giants) in a form that everybody can access, modify and further expand.

3.1. Software architecture

The architecture of the library is a simple stack (see Fig 5).

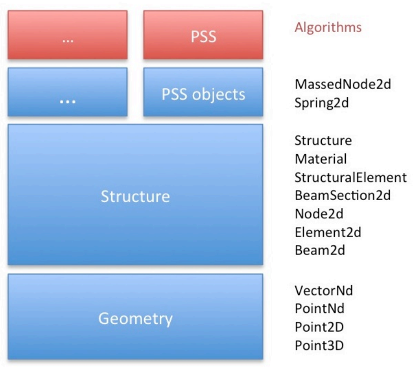


Fig 5. Engineering-open stack design with some of the object classes listed on the left hand side.

The bases of a number of important object classes are geometrical classes, which are organized in the "geometry" layer of the stack. The next level in the stack, the "structure" layer, contains a number of structural objects which can be further extended to be used by the algorithm-specific object layers and the algorithms themselves. In Fig 5 the particle-spring-system stack is shown which add MassedNode2d and Spring2d as algorithm-specific object classes, which are utilized by the PSS algorithm in the "algorithm" layer.

3.2. Implementation

Current developments are based on the Python programming language [8] so that the code can be used cross-platform (e.g. Windows, Mac OS and Linux operating systems are supported). The Python ecosystem has a wide variety of tools to build a many different applications based on engineering-open.

The current implementation is limited to the first developments and does not contain a lot of complicated algorithms yet. Currently, the geometry layer contains simple classes for a n-dimensional vectors of which n-dimensional points are derived. From these n-dimensional points the two-dimensional and three-dimensional point are derived. In the structure layer simple structural classes have been implemented, such as the Structure class, which collects all StructuralElements into a structure to be used by the algorithms. Furthermore, base classes such as Material, two-dimensional beam sections, two-dimensional nodes, elements and beams have been implemented.

Based on these object classes 'Calculations' and 'Algorithms' have been introduced. A calculation performs a simple calculation while an algorithm performs a more complex routine to come to an answer.

An example of a 2D calculation, which has been implemented, is a simply supported beam, which based on a load gives the maximum shear force, maximum bending moment and maximum deflection of the beam based on simple rules of thumb.

An example of an algorithm, which has been implemented, is a 2D particle spring system. This algorithm uses step-wise damped motion in small time increments based on simple laws of mass (force is mass times acceleration) and motion (e.g. the change is velocity is acceleration) to calculate an equilibrium state of a system of linear springs and masses. This algorithm can for example be used for form finding of cables and cable-nets.

It must be noted that this algorithm is not ideal for these use cases because the algorithm struggles with convergence and stability problems, but it provides a simple test case for the library. At a later stage more complicated and more functional algorithms can be added to perform the real analysis.

Unit testing making use of Python's internal framework has been used to verify the functionality of the library. The source code is hosted on Github (http://github.com/jeroencoenders/engineering-open). For more information on the implementation, refer to the source code in this repository.

Note that the current implementation is only a library (a collection of Python modules) and therefore does not contain any (graphical) user interface. There is no intention to add these to the library as users are encouraged to develop their own user interfaces (and share these with the rest of the world).

3.3. Licensing

As a license the GNU Public License version 3.0 (GPL v3) has been utilized which provides the source code open source for a variety of purposes, including educational and commercial purposed. However, one of the main restrictions is that derivative developments will have to be shared back to the original initiative (copy-left). In this way the initiative guarantees continuous improvement of the software code and the fact that the source code will remain open to everybody.

3.4. Future developments

The current implementation of engineering-open is rather limited as this initiative is in its early stages and has not attracted a lot of contributors yet. However, plans exist to expand the library with a large number of calculations and algorithms.

Below a list of some examples of new directions are given on which some work has already started:

- Current implementations are mostly limited to two-dimensional space. The plan is to extend all calculations and algorithms to three-dimensional space.
- A Finite Element Method (FEM) solver will be added to library to solve simple (linear) beam structures in two-dimensional and three-dimensional space. In the future this solver can be extended to more complicated elements, such as e.g. membrane and shell elements.
- A number of form finding methods, such as the Force Density Method [9] and Dynamic Relaxation [10], will be added to the library to address cable-net and membrane structures.
- Meshing algorithms will be added to the library.
- A number of structural optimisation methods, such Genetic Algorithms [11], will be added in order to being able to quickly optimize structures.

Most calculations and algorithms are aimed at structural engineering. However, the approach is open for other engineering disciplines to contribute.

4. DISCUSSION

This paper has presented two open source development initiative in the field of design and engineering which deliberately share their software code of with the rest of the world in order to stimulate verification, validation, benchmarking, continuous improvement and innovation by removing the barriers of the source code of software not being available.

Both developments are still at early stage with a limited number of collaborators and contributors and still have to build a larger community of users. The author encourages downloading and evaluating the software and giving feedback through the Github repository. The author would also like to encourage colleagues from academia and practice to share their knowledge by contributing code to be used by others.

5. CONCLUSIONS

In this paper two open source initiatives have been discussed which aim to make source code available to designers, engineers, researchers, developers, scientists and students. The advantages of making this source code open to everybody are numerous, amongst others: no more ‘black boxes’, verification, validation and benchmarking directly available and inspiration to innovate by building by ‘standing on the shoulders of giants’ – the current state of the art.

6. ACKNOWLEDGEMENTS

The author would like to express his gratitude to Tata Steel, Bouwen met Staal (the Dutch Steel institute) and Arup for their support of the initial steps of the sustainability-open research and development project.

7. REFERENCES

[1] *sustainability-open website*, http://www.sustainability-open.com, accessed 17 February 2013.

[2] *engineering-open website*, http://www.engineering-open.com, accessed 17 February 2013.

[3] Coenders J.L., Barriers in computational structural design, Journal of the International Association of Shell and Spatial Structures, 48(4) (2007), pp. 51-62.

[4] Microsoft, *.NET technology website*, accessed through http://www.microsoft.com/net on 30 April 2013.

[5] Github Inc, *Github website*, accessed through http://www.github.com on 30 April 2013.

[6] Davidson S, *Grasshopper website*, http://www.grasshopper3d.com, accessed 17 February 2013.

[7] Eastman, C, *Building Product Models: Computer Environments Supporting Design and Construction*, CRCPress, 1999.

[8] *Python website*, http://www.python.org, accessed 17 February 2013.

[9] Schek, HJ, *The force density method for form finding and computation of general networks*, Computer Methods in Applied Mechanics and Engineering, Volume 3, Issue 1, January (1974), pp. 115‑134.

[10] Barnes, MR, *Form-finding and analysis of prestressed nets and membranes,* Computers & Structures, Volume 30, Issue 3 (1988), pp. 685‑695.

[11] Michell, M, *An Introduction to Genetic Algorithms*, MITPress, 2nd edition (1996).