Generating Early Design Alternatives Based on Formalized Requirements and Geospatial Data

Steffen Hempel, steffen.hempel@kit.edu
Karlsruhe Institute of Technology, Germany

Joachim Benner, joachim.benner@kit.edu
Karlsruhe Institute of Technology, Germany

Karl-Heinz Häfele, karl-heinz.haefele@kit.edu
Karlsruhe Institute of Technology, Germany

Abstract
In designing new buildings, a number of constraints need to be considered. Among these are requirements concerning the overall building size as well as the number, function, and size of the needed spaces. Additional boundary conditions are implied by the neighborhood of the planned building (e.g. existing buildings and infrastructure, natural environment). In state-of-the-art design processes, these constraints mostly are taken into account only formally. This paper proposes to formalize the space requirements and use a geospatial data model of the surrounding area in order to easily generate different design alternatives in the early design phase.
Such approach has a number of advantages. Based on a machine-readable description of the space requirements, different alternatives for a simplified building envelope and a basic space layout can be generated semi-automatically. At least on large sites like hospital districts or industrial sites, different alternatives for the location of a new building may exist. By using geospatial data, it can be checked whether a specific design variant can be realized at a certain location. This will support the decision-making process in finding the most promising early design model, which is promoted in the following design steps.
This paper describes a software prototype using a proof-of-concept algorithm to generate several first drafts of building floor plans. These are enriched with additional data like placement or information about volume and used space. Finally, the floor plans are extruded to volumetric objects and exported into the IFC format.

Keywords: early design variants, ifc, space requirements, knowledge-based design

1 Introduction
Planning a new building that is integrated in an existing neighborhood of functionally related buildings requires consideration not only of the building level, but also of the site level. Examples of such sites are health care centers, industrial facilities or research and university campuses.

During the early design phase of a building, different factors like cost or energy-efficiency need to be considered. A decision towards a specific design alternative has a large impact on the whole building life cycle. The earlier a decision for a suitable alternative is made, the easier can it be implemented, and, therefore, the cost of the following stages are reduced. On the other hand, subsequent changes to the design may result in increased costs, are very time-consuming, difficult to implement, and may lead to quality issues. The further a project has advanced in its life cycle, the less the possibilities of influencing impacts and cost (Kohler & Moffatt, 2003). The proposed approach to avoiding these downsides is to generate many design alternatives based on formal requirements as concrete proposals in an early phase. A proposal contains the floor plan of the entire building, consisting of the outlines of all stories and the room layout within a story outline. These proposals can be rated and the most promising ones are developed further.

The Early Design Configurator presented in this paper is a tool for semi-automatically generating proposals which fulfill the formalized requirements in the best possible way. It is part of

the STREAMER project (Streamer, 2013). In this project, the Early Design Configurator is embedded in a toolchain where additional tools are provided by the STREAMER partners. Among others, these tools generate the requirements used by the Early Design Configurator and use the proposals generated by the Early Design Configurator to calculate Key Performance Indicators. These Key Performance Indicators, in the case of STREAMER, "inform owners on how to make dramatic increases in energy performance" (ARUP, 2014). Although STREAMER concentrates on mixed-use health care districts (Benner, et al., 2015), the solution presented in this paper is also generally applicable.

2 State of the Art

The topic of automatic layout generation is very old (Schneider, et al., 2010). (Schneider, et al., 2010) divide the problem of computer-aided layout generation into operational and non-operational parts. The operational part can be solved algorithmically, while non-operational problems are dependent on human interaction.

Whereas most approaches presented seem to focus on the non-operational part supporting the creative design, the approach of the Early Design Configurator concentrates on the non-operational parts in favor of optimizing the operational and therefore measureable parts. The focus is laid on optimizing a floor plan towards a functional goal.

Furthermore, (Schneider, et al., 2010) give an overview of how automated layout design has been tackled so far with "cellular automata and agent-based systems", "shape grammars", "physically-based systems", and "evolutionary algorithms". The conclusion is, that "it appears that the combination of generative processes with evolutionary methods is the most promising approach".

Examples of the use of evolutionary algorithms for layout design based on a rectangular grid have been presented by (Jo & Gero, 1998) and (Rosenman, 1996). Even though the reliance on a rectangular grid is criticized by (Schneider, et al., 2010) because "the possible geometries are very limited", the use of a rectangular grid may be considered sufficient for early designs.

Another approach, focused on creating realistic floor plans, is presented by (Marson & Raupp, 2010). This approach is based on computation speed and therefore may allow solving small layout generation problems encountered in a large functional optimization process. The constraints on different rooms are of course less strict, because visual representation is more important. The generated layout of the floor plan is also more concentrated on personal housing and less on the more structured layout of industrial complexes.

Project Akaba (Autodesk, 2015) is generating 3D buildings without a previously defined outer shell for an early design stage. It supports different generation algorithms (Random, Cellular Automata, L-System Grammars) and is able to optimize towards different priorities (e.g. usage, layout, site and minimize circulation). As input the required rooms and their relations can be defined.

Onuma (Onuma Inc., 2013) among others offers a solution for rapid prototyping. This approach allows the creation of floor plans heeding previously defined requirements giving the user visual indicators of the current quality of the layout. Rapid prototyping approaches may offer the possibility of designing floor plans based on defined restrictions. This may go from simple area to more complex constraints. Still, varying requirements from different optimization goals require a time-consuming manual creation of different designs. Also the extension with additional constraints may prove to be difficult.

3 Solution Approach

To use the requirements, these have to fulfill certain criteria. Chapter 3.1 shows how requirements prior to the generation of early designs have to be defined in a formalized way. Generation of early designs requires a defined basic workflow for this stage (see Chapter 3.2). A broader view of how the planning of early design is embedded within a global system architecture is show in Chapter 3.3.

3.1 Formalizing Requirements

The requirements are divided into two groups, project-specific brief requirements, and general non-project-specific knowledge.

The basic goal of the brief requirements is to have a list of classified rooms with a count of how many of these rooms are required for the project. Additionally, there are semantic labels which describe a room further e.g.,

- a patient room needs a specific temperature,
- a patient toilet is in the hotel zone,
- an office needs a specified area,
- an operating room is in a clean zone.

The knowledge describes the relations between different classifications of rooms. These relations between classifications can also be derived from the attribute of a room. Assuming there is a rule "rooms with the same temperature requirements should be close to each other", the reference to the specific object is not given by its classification name (e.g. "Patient Room"), but by comparing the attribute "temperature" of two objects. When this comparison is resolved, the rule can be described by referencing classifications directly again: The objects "Patient room" and "Patient toilet" have the same "temperature" results in the rule "Patient rooms and patient toilets should be close to each other".

Most relations are clustering rules e.g.,

- rooms of the same temperature should be clustered,
- rooms of the same hygiene level should be connected,
- five-patient rooms need at least one toilet.

There are several expectations concerning requirements. All of the requirements need to be translatable into geometric restrictions:

- A room needs a minimum width/height,
- a room needs an area,
- minimum/maximum distance between rooms,
- line of sight between rooms,
- maximum walking distance between rooms,
- minimum/maximum area of a room,
- a room requires daylight/no daylight/maybe daylight,
- a room should be close to another room,
- rooms need to be in the same zone,
- rooms need to be on the same floor,
- minimum of N rooms of classification X at maximum distance.

These restrictions are always measurable with either simple comparisons (same zone/floor) or with more specialized measures like Euclidean distance (minimum/maximum distance) up to specialized algorithms like path finding in a generated graph of the ways between rooms (minimum/maximum walking distance). Additional restrictions may need to be introduced when needed.

3.2 Early Design Workflow

There are many possible workflows leading to proposals for an early design. Therefore, the workflow shown in Figure 1 was developed as a prototypic workflow for the Early Design Configurator.

The first step is the import of the requirements. When all requirements are set, a building shell is created. This outside-in approach, defining the outer shell first and then filling it with rooms according to the requirements, accommodates the fact that buildings are created either with a specific location in mind or that they are extensions to already existing districts or other buildings. The building shell sets additional boundary conditions for the previously defined requirements e.g., the area of defined rooms must not be larger than the area provided by the building shell.

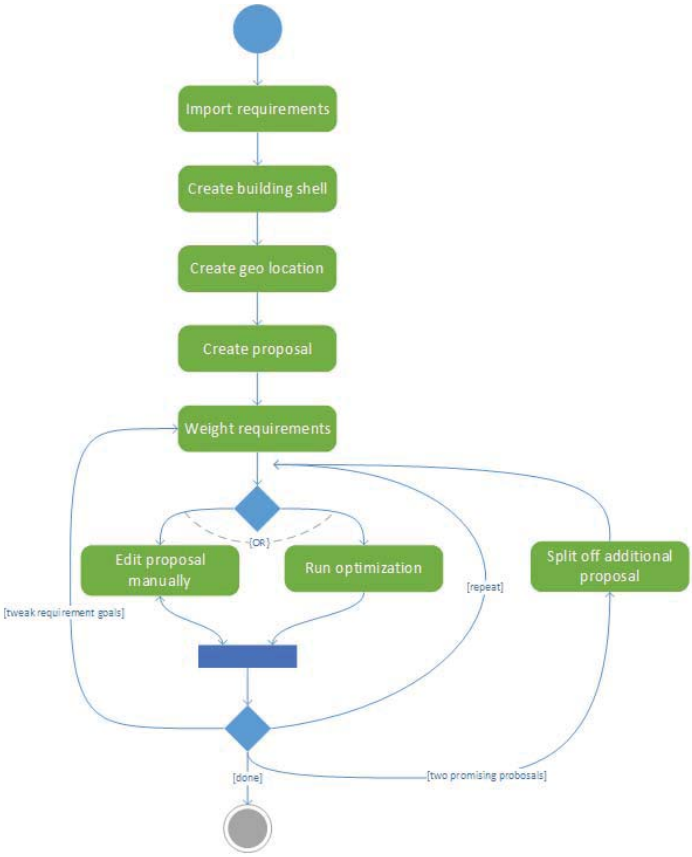

Figure 1 Early Design Configurator workflow

Utilizing the outer shell of the building, the building location and orientation can be defined and validated, using raster data from map services.

From this point on, the automatic generation of proposals starts. Each proposal is a floor plan variation fitted into the building shell created from the aforementioned requirements.

The simplest way to create different proposals is the variation of the building shell or even the requirements themselves. The more important possibility is changing the weight of requirements, which results in different layouts. These weights are defined before creating a new proposal.

It is also possible for the designer to influence the layout by editing the proposal manually. This is done by locking rooms into positions or manually selecting a layout template. These manual changes may result in invalid layouts or layouts which are worse in how the requirements have been fulfilled. This may either be intentional, or the automatic generation algorithm can be run again to fix the floor plan, but this time heeding the changes made by the designer. Also, after these manual changes are made, additional proposals may be split off to keep both for consideration.

3.3 Technical Interfaces

The Early Design Configurator must be embedded in a global system architecture. The connection between other software and the Early Design Configurator is shown in Figure 2.

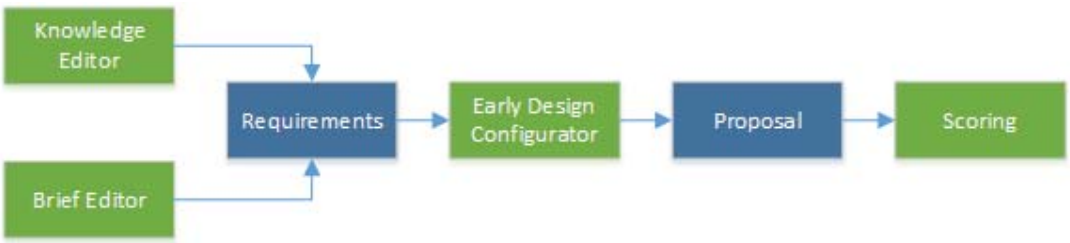

Figure 2 Early Design Configurator interfaces

The first two parallel steps are the creation of the project-specific brief requirements in the brief editor and the knowledge in a knowledge editor by the customer and experts, respectively. The brief requirements are defined in commercial software like BriefBuilder or dRofus and then exported as IFC or CSV files. The knowledge is defined in an OWL ontology which is created with a domain-specific knowledge editor. Both, knowledge and requirements are imported into the Early Design Configurator as requirements.

Proposals generated in the Early Design Configurator from the requirements are exported as early design proposals. The chosen export format is IFC (ISO 16739, 2013), because many tools on the market used in the architectural sector support it (buildingSMART International Ltd., 2015). Another format that may be used for energy simulation is gbXML.

The exported proposal is scored according to the defined Key Performance Indicators with tools developed in the STREAMER project and then further developed with other tools.

4 Early Design Configurator

The Early Design Configurator is a first approach to developing a software prototype for the process described in Chapter 3.1 and is part of the Streamer project. It uses an evolutionary algorithm to generate a floor plan optimized towards the specified requirements.

As input, brief requirements and knowledge are imported into the Early Design Configurator. In the Early Design Configurator, building shape (see Chapter 4.1) and additional geospatial data (Chapter 4.2) are created.

All this information in varied combinations is used in the layouting process to generate proposals for an early design (Chapters 4.3 and 4.4). The resulting proposals can then be exported into IFC for further processing.

4.1 Defining Building Shells

The building is defined as connected 2D rectangles called shapes to define the outer shell. The connected shapes define the outline of the building. Each shape additionally has a level count and a starting level. The building shell can be generated using the Building Editor of the Early Design Configurator shown in Figure 3 (top). The image represents a bridge building – two towers left and right, and the bridge in the middle. The whole building has two floors. The resulting 3D representation is also shown in Figure 3 (bottom).

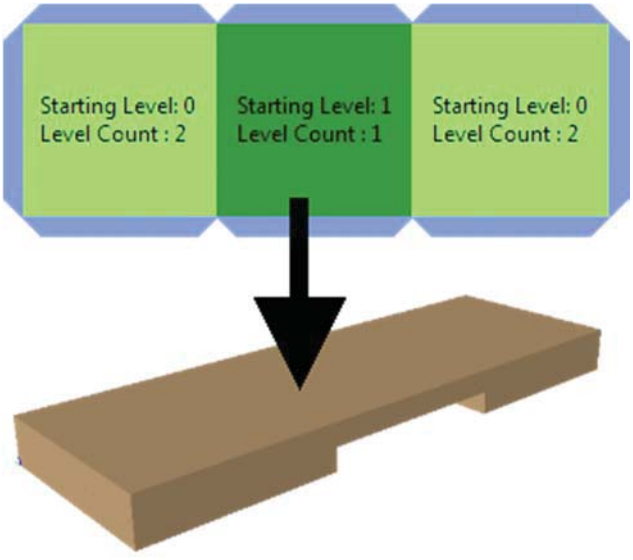

Figure 3 Building representation in the Building Editor

4.2 Integration of GIS Data

Once a building shell has been created, its geolocation can be defined by placing it on a map.

The data for the display of the current surroundings of the building are raster images taken from an OpenStreetMap tile server. Figure 4 shows the building defined in Chapter 4.1 placed in the Early Design Configurator Site Editor.

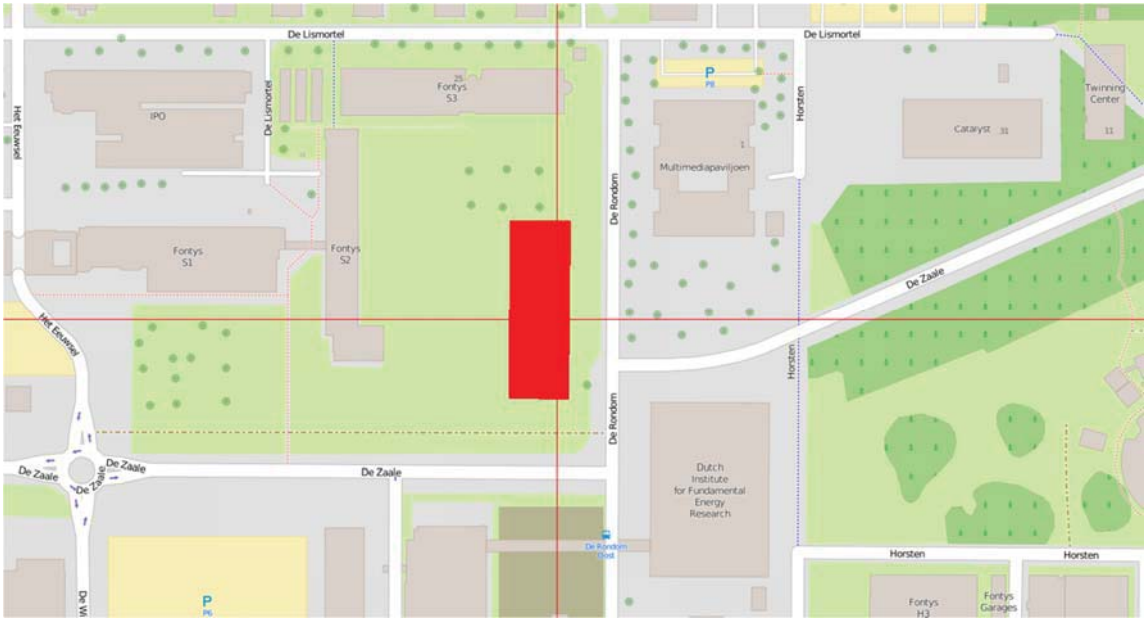

Figure 4 Building placement with the Site Editor

4.3 Constraints

Constraints are a central concept of the generation algorithm. They can be used to calculate the rating of a building according to the defined requirements, if these requirements satisfy the restriction defined in Chapter 3.1. The rating is called satisfaction and is a value from 0 (best) to ∞ (worst).

As the satisfaction from different constraints is used to calculate the overall satisfaction of the whole layout process, the satisfaction value from one constraint should go from 0 (best) to 1 (worst). A number greater than 1 is used to signal special cases where a manipulation operation to the layout resulted in a violation of hard constraints, therefore resulting in an invalid layout.

There are three basic types of constraints: Hard, soft and combined constraints. A hard constraint is a Boolean condition. This means that a violation of this constraint results in a bad satisfaction value.

$$f(x) = \begin{cases} 0, & if\ satisfied \\ bad\_satisfaction, & else \end{cases} \quad (1)$$

As shown in Equation 1, this bad satisfaction value may be either 1 or, for cases where a layout should have a higher probability to be discarded if the constraint is violated, an even higher value. Examples of these kinds of constraints are:

- Room A needs to be within 20 meters of room B,
- the layout must not be larger than the building shape,
- corridor A must connect to corridor B.

A soft constraint results in a satisfaction that gets increasingly worse the more the satisfaction is violated (Equation 2). This kind of constraint requires a border value, which in most cases is the maximum value for the measured value.

$$satisfaction_{soft} = \frac{actual\_value}{border\_value} \quad (2)$$

In special cases, where the border value is not the maximum value, the satisfaction may be above 1. Examples of these kinds of constraints are:

- Room A needs to be as close to room B as possible,
- the walking distance between room A and room B must be as small as possible.

Another case combines both satisfaction calculation methods (Equation 3) such that either a soft constraint is used inside the range of the border value or the result of the constraint is a bad satisfaction.

$$satisfaction_{combined} = \begin{cases} \frac{actual\_value}{border\_value}, & if\ in\ range \\ bad\_satisfaction, & else \end{cases} \quad (3)$$

This is an example of the following cases:

- Room A needs to be at least within 10 meters of room B.
- There need to be at least 5 rooms of type C within N meters of a room A.

There are currently only built-in constraints which enforce boundary conditions implemented in the Early Design Configurator and no constraints are taken from the actual requirements. The defined constraints are:

- *WidthConstraint* and *HeightConstraint* are combined constraints which ensure that the layout is not larger than the building shape.
- *WastedSpaceConstraint* is a soft constraint which gets better; the less unused space is in the current layout.
- *CorridorConstraint* is a hard constraint which is bad if the corridors on the borders of building shapes do not connect.

Also built-in is how the Early Design Configurator handles the count, size, and area of a room. These are actual values taken from imported requirements. They are not implemented as constraints but are taken into account when placing a room in the layout of a proposal.

4.4 Generating Layouts

An evolutionary algorithm is used to generate the layouts. For each part of an empty building shell, a first layout is randomly chosen from a database of layout templates. These layout templates define how corridors are to be placed, and how the rooms are aligned along these corridors in strings of rooms.

The algorithm runs in iterations. Three operations are possible as mutations:

1) The layout is randomly filled with rooms – either a room is added or removed.
2) Rooms are resized randomly.
3) The layout is changed to another random layout.

The resulting floor plan is rated by calculating in Equation 4 the satisfaction from all constraints, which is then compared to previous floor plans.

$$satisfaction_{global} = \sum_{i=0}^{n} satisfaction(constraint_i) \times importance_i \quad (4)$$

Different constraints may need to have their satisfaction value weighted according to the importance of the constraint. This allows defining goals by preferring constraints which accommodate the goal. The best floor plan is retained and used for a next iteration where the whole process is repeated.

As evolutionary algorithm, the resulting layout may never be the best possible layout. Also, there exists no defined termination state, so the algorithm may run either for a defined time or a defined number of iterations. However, the current implementation allows the calculation of the global satisfaction to be extended by defining new constraints which integrate additional requirements.

5 Examples

The following example uses the bridge building already shown in Chapter 4.1. In Figure 5, parts of the brief requirements are shown. These are randomly generated test requirements. The column “Name” is the classification of the requirement. “Size” is a hint as to how the size of this room normally is. “Area” denotes the actual required area of the room, and “Count” stands for how many of these rooms are required.

Name	Size	Area	Count
Test 0	4.00 x 5.00	20.00	3
Test 1	1.00 x 4.00	4.00	6
Test 2	2.00 x 4.00	8.00	2
Test 3	1.00 x 1.00	1.00	1
Test 4	2.00 x 3.00	6.00	1
Test 5	1.00 x 2.00	2.00	6
Test 6	2.00 x 4.00	8.00	9
Test 7	2.00 x 4.00	8.00	10
Test 8	4.00 x 5.00	20.00	4

Figure 5 Randomly generated requirements

The proposal generated from the requirements and outer shell is 2D and can be shown in the Early Design Configurator. The two base towers of the first floor are shown in Figure 6. The bridge surface is show in Figure 7. In both figures, the blue rectangles are the placed rooms. The yellow and green lines show the borders of layout parts. In Figure 7, the layout below the current floor is shown in a lighter color. This layout part is not valid, because no constraint is implemented that checks if all rooms are accessible.

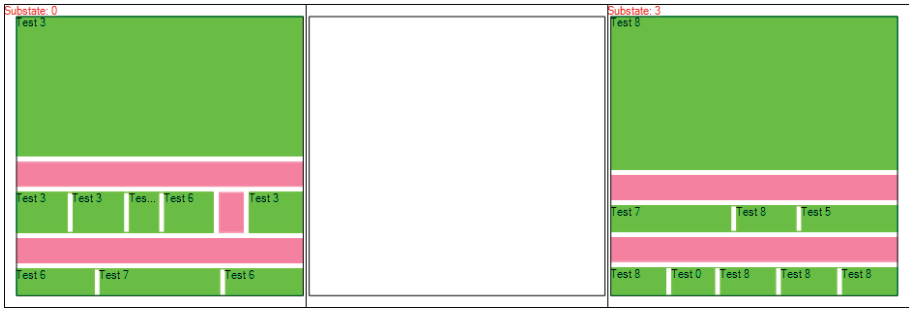

Figure 6 Floor plan of the first floor

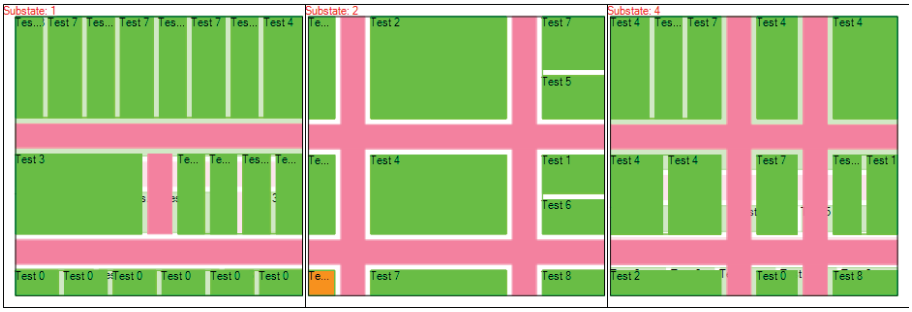

Figure 7 Floor plan of the second floor

The proposal exported into an IFC file can be shown in a software like IFCExplorer. A screenshot of the exported proposal is displayed in Figure 8. It shows the outer shell of the building, and the rooms placed inside the building.

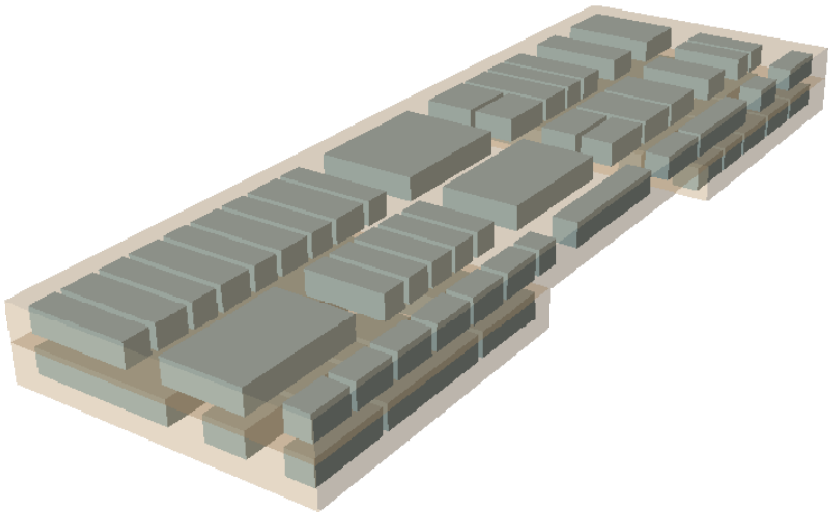

Figure 8 Rooms of the building

6 Conclusion and Outlook

The developed prototype demonstrates the concepts and potential of the Early Design Configurator. With a focus on different goals, it is possible to generate many proposals very fast.

A workflow has been defined, which lays the basis of how the Early Design Configurator is used, how many alternatives for proposals are made in the automatic process, and how they can be manipulated to generate additional variations.

The Early Design Configurator workflow is embedded in previous and following phases. The results from previous phases comprise requirements and knowledge. Brief requirements are imported from commercial tools in the form of CSV files. Currently, the supported requirements on room level are base area and classification. It has been discussed which additional requirements need to be supported.

The support of other export formats in addition to IFC and the generation of enriched IFC files are required to support tools in subsequent design phases. The addition of walls, slabs and ceilings, and the support of creating the spaces for corridors will be the first steps in extending the Early Design Configurator. Furthermore, export formats like gbXML or CityGML may be helpful for interfacing with energy simulation tools or GIS systems, respectively.

A building shell can be generated and its location can be defined through Building Editor and Site Editor. Extraction of additional information like distances to points of interest could be taken from GIS data containing semantic information, like the location of the nearest public transport station. Regarding the topic of energy efficiency, the simulation of shading from adjacent buildings and the influence of solar radiation may be taken into account. Also, on a very speculative basis, things like the view from the building may be considered by evaluating available GIS data.

The constraints have been described as an algorithmic representation for the formalized requirements. Applying the currently implemented constraints with the presented version of the algorithm already results in very basic but already presentable proposals. To support the creation of valid layouts, the import of more complex requirements must be implemented. This also includes the implementation of the matching constraints. Additionally, there are more boundary condition constraints (see Chapter 4.3) which need to be added, enforcing the accessibility of rooms through the corridors, and the vertical connection of lifts and staircases.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme under grant agreement No. 608739 (Project STREAMER).

References

ARUP. (2014). *Building-orientated EeB KPIs of newly designed and retrofitted buildings.* Retrieved June 29, 2015, from: http://www.streamer-project.eu/Downloads/D3.1.pdf.

Benner, J., Häfele, K.-H., Bonsma, P., Bourdeau, M., Soubra, S., Sleiman, H., & Robert, S. (2015). Interoperable tools for designing energy-efficient buildings in healthcare districts. In A. Mahdavi, B. Martens, & R. Scherer (Eds.), *ECPPM 2014* (pp. 915-922). Wien: CRC Press.

buildingSMART International Ltd. (2015). *Participants of the official buildingSMART IFC2x3 Coordination View V2.0 certification process.* Retrieved 05 11, 2015, from http://www.buildingsmart-tech.org/certification/ifc-certification-2.0/ifc2x3-cv-v2.0-certification/participants

ISO 16739:2013 Industry Foundation Classes (IFC) for data sharing in the construction and fa-cility management industries. (2013).

Jo, J. H., & Gero, J. S. (1998). Space layout planning using an evolutionary approach. In *Artificial Intelligence in Engineering* (pp. 149-162).

Kohler, N., & Moffatt, S. (2003, April-September). Life-cycle analysis of the built environment. *UNEP Industry and Environment*, pp. 17-21.

Marson, F., & Raupp, S. (2010). Automatic Real-Time Generation of Floor Plans Based on Squarified Treemaps Algorithm.

Autodesk (2015), Project Akaba [Software], available from Autodesk on request

Onuma Inc. (2013). *Onuma Inc.* Retrieved 4 23, 2015, from https://www.onuma.com/

Rosenman, M. A. (1996). The Generation of Form Using an Evolutionary Approach. In *Artificial Intelligence in Design '96* (pp. 643-662).

Schneider, S., Fischer, J.-R., & König, R. (2010). Rethinking Automated Layout Design: Developing a Creative Evolutionary Design Method for the Layout Problems in Architecture and Urban Design. *Design Computing and Cognition DCC'10. J.S Gero (ed)*, pp. 367-386.

Streamer. (2013). *Streamer - European research on energy-efficient healthcare districts.* Retrieved May 8, 2015, from http://www.streamer-project.eu/