



A Generative Design Method Based on a Graph Transformation System

Grażyna Ślusarczyk^(✉) , Barbara Strug^{}, and Ewa Grabska^{}

Institute of Applied Computer Science, Jagiellonian University,
Lojasiewicza 11, 30-059 Kraków, Poland
{grazyna.slusarczyk, barbara.strug, ewa.grabska}@uj.edu.pl

Abstract. This paper deals with a Computer-Aided Design (CAD) tool supporting the conceptual phase of the design process. Design drawings are created by the designer with the use of a visual editor. It is possible to interactively define design requirements and/or visual design patterns, which are important features for expected/required characteristics of the task solution. Design drawings, requirements, and specified visual patterns have their internal graph representations, which are automatically obtained during the design process. Based on design actions taken by the designer during the design process, graph transformation rules, which are used to derive graphs representing design objects, are constructed. Additional graph transformation rules are created on the ground of design patterns and requirements specified by the designer. The obtained set of graph transformation rules, called a graph transformation system, is used to automatically generate graphs corresponding to new objects. The approach is illustrated by examples of designing indoor swimming pools.

Keywords: Generative design · Design patterns · Graph structures · Graph transformation rules

1 Introduction

This paper deals with a Computer-Aided Design (CAD) tool supporting the conceptual phase of the design process by inferring graph transformation rules from initial designs, specified requirements, and design patterns. Nowadays CAD systems are widely used to support the design process by facilitating integration of its various phases and aspects [9]. However, they still lack generative methods to easily generate classes of design variants. Many models of design objects often have similar structures and differ only in sizes or the way of fitting to varying environmental conditions [2, 10].

Design is seen as a creative task because novel design problems typically are open-ended and ill-defined. Therefore CAD systems should support the creative phase of the design process by enabling the designer to experiment with candidate solutions. The proposed CAD-like interactive environment allows for generating many candidate solutions with similar structures but different geometry and parameters based on the small set of possible solutions created by the

designer. This type of support produces new knowledge that the designer can take into account in refining or reconsidering candidate solutions [3].

The use of graphs and graph grammars to represent and generate structures of designed objects in the initial phase of design has taken place in CAD systems since the eighties of the last century. Currently, domain-specific visual language editors encourage designers to create their initial design solutions in the form of design drawings, while graphs are internal representations of these drawings and make it possible to support the designer's actions by a CAD system. During the design process, each sequence of steps of creating a design drawing by the designer can be automatically transformed into a graph transformation rule describing the introduced changes.

The proposed approach is illustrated by examples of designing indoor swimming pools, where the designer generates design drawings using a domain-specific visual editor. In the case of designing a floor layout of a swimming pool, the designer starts by drawing an outline of a building and then divides it into areas, which are recursively divided until the level of individual rooms is reached. In this way, the designer ends with a complete floor layout showing sizes and positions of all rooms. During the design process, the architect can interactively select parts of a created design and designate them as important features for expected/required object characteristics or specify some requirements using a set of dialog windows.

In the presented approach both topological structures and semantic properties (geometrical parameters, materials, environment aspects) of designs are represented by composition graphs (CP-graphs) and their attributes, respectively [1]. A CP-graph is a labelled and attributed graph, where nodes represent components of artefacts and are labelled by names of components they represent. To each node, a number of bonds representing fragments of components and expressing potential connections between components is assigned. Each edge connects bonds of two different nodes and represents relations between fragments of components represented by these nodes and is labelled by the relation names. To nodes and bonds attributes specifying properties of components and their parts are assigned [4]. The whole design drawings generated by the designer, the specified parts (patterns), and requirements are automatically transformed into the corresponding CP-graphs.

CP-graphs representing designed objects can be generated and modified using composition graph transformation rules. Some rules of a graph transformation system are constructed on the basis of design actions taken by the designer during the design process. Additional graph transformation rules are created on the ground of design patterns and requirements specified by the designer. The obtained set of graph transformation rules is used to automatically generate CP-graphs corresponding to new objects with geometry and material properties specified by graph attributes. The possibility of relating attributes of right-hand sides of CP-graph rules to attributes of their left-hand sides enables the system to capture parametric modeling knowledge. CP-graph transformation rules are

designed in the unified graphical environment delivered by the system called *GraphTool*, which supports graph transformations [11].

This paper contributes to the field of computational methods by supporting the design of a graph transformation system. Graph transformation rules are typically difficult to construct even for people with considerable background knowledge, especially when the characteristic features and design requirements should be taken into account. In our approach, graph transformation rules are inferred based on design actions taken by the designer and, design patterns and requirements specified by the designer. The proposed generative method allows the system supporting design to automatically model alternative object layouts, which can be easily adapted to different use case scenarios and environmental conditions.

2 Generating Design Drawings

In the initial stage of the design process, understanding of requirements is often associated with visualizations of early conceptual solutions. In our approach, the system supporting building design [5] enables the designer to draw building layouts with the use of a visual editor. It is also possible to interactively define design requirements and/or visual design patterns, which are important features for expected/required characteristics of the task solution. Moreover, visualizations of functional artefact models can be developed to help organise research.

In the following, the systems supporting design [5], which enable the designer to draw building layouts will be considered.

Example 1. Let us consider an example of designing an indoor swimming pool. At first, a 2D outline of the building is drawn in an orthogonal grid (Fig. 1a). In the next step, taking into consideration the type of the swimming pool (recreational, sports, learner) and an approximate number of users, the designer

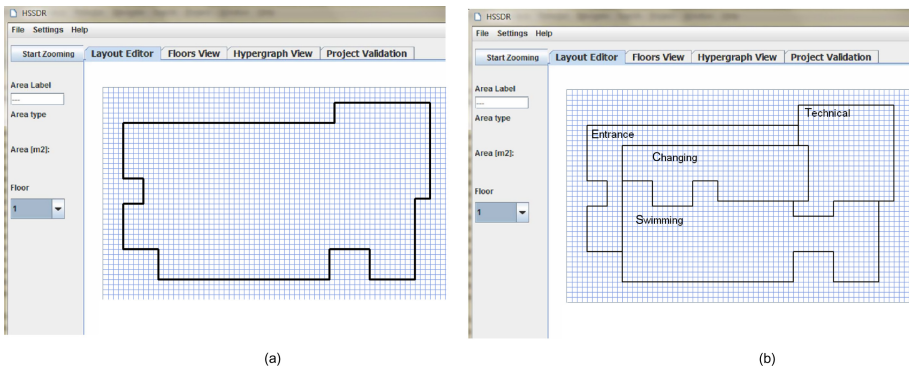


Fig. 1. a) The outline of the swimming pool b) the decomposition of the whole area into four functional areas.

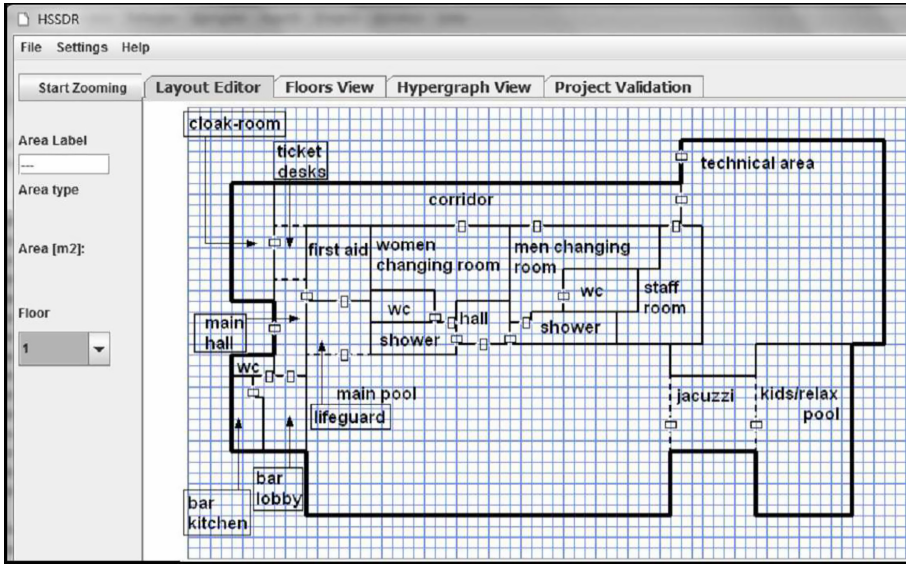


Fig. 2. The floor layout of the swimming pool.

decomposes the area determined by this contour into functional areas (Fig. 1b). Then functional areas, namely the *entrance area*, *swimming area*, and *changing area*, are decomposed into appropriate rooms. In the entrance area the *main hall*, the *ticket desks*, the *cloakroom*, the *corridor*, the *toilets*, the *bar lobby*, and the *bar kitchen* are distinguished. The changing area is decomposed into *women* and *men changing rooms*, the *toilets*, the *showers*, the *staff room*, and the *first aid room*. The swimming area is divided into the *main pool*, the *jacuzzi*, the *kids' pool*, the *lifeguard room*, and the *hall*. It is assumed that in successive design steps the designer gives labels starting from capital letters to areas that will be further divided, while labels starting from small letters are given to terminal spaces. At the last step the small rectangles representing *accessibility* relations between rooms are added and some continuous lines shared by polygons and denoting the *adjacency* relations between rooms are changed into the dashed ones, which represent the *visibility* relations between them. The whole obtained floor layout is shown in Fig. 2.

The designer has also the possibility to specify some requirements using a set of dialog windows. For example the functional areas which should be presented in the design drawing can be chosen. Each of the four areas (namely the *communication/entrance area*, *swimming area*, *changing area*, and *technical area*) together with their orientation can be selected. For each of these areas, the designer can further select rooms which it should contain. The selection process of functional areas and elements of the swimming area is shown in Fig. 3.



Fig. 3. Selection of functional areas and component parts of the swimming area.

Another way of specifying layout characteristic features is by indicating visual patterns of the designed drawing. In Fig. 4, which presents the layout of the first floor of a swimming pool, the visibility relation between the restaurant and the main swimming pool is indicated by the red ellipse, as the one which is important for the designer.

3 CP-Graph Representation of Designed Objects

In our approach the design drawings of building layouts are automatically transformed into their internal representations in the form of composition graphs (CP-graphs) [5–8]. A CP-graph is a labelled and attributed graph, where nodes

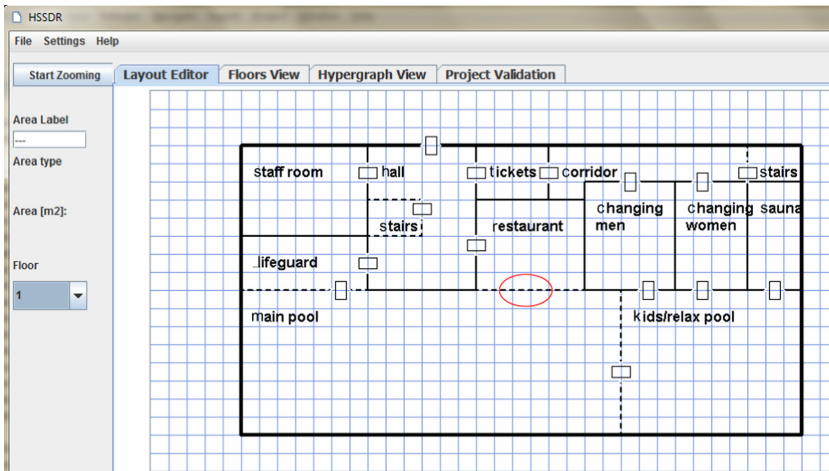


Fig. 4. Indicating the visibility relation between the restaurant and the main swimming pool.

represent components of the object being designed and are labelled by names of components they represent. To each node a number of bonds expressing potential connections between components is assigned. Nodes and bonds have attributes specifying properties of components and their parts, assigned to them, respectively [4].

Each edge of a CP-graph connects bonds of two different nodes, and it represents a relation between parts of the components represented by these bonds and is labelled by the relation name. The whole design drawings generated by the designer, the specified parts (patterns) and requirements, are automatically transformed into the corresponding CP-graphs.

The formal definition of a CP-graph is as follows.

Let Σ be a finite alphabet used to label object nodes, bond nodes and edges. Let A be a nonempty, finite set of attributes. For each attribute $a \in A$, let D_a be a fixed, nonempty set of its admissible values, known as the domain of a .

A CP-graph G is a tuple $G = (V, B, E, bd, s, t, lab, atr)$, where:

- V, B, E are pairwise disjoint finite sets, whose elements are respectively called nodes, bonds and edges,
- $bd : V \rightarrow 2^B$ is a function assigning sets of bonds to nodes in such a way that $\forall x \in B \exists! y \in V : x \in bd(y)$, i.e., each bond belongs to exactly one node,
- $s, t : E \rightarrow B$ are functions assigning to edges source and target bonds, respectively, in such a way that $\forall e \in E \exists x, y \in V : s(e) \in bd(x) \wedge t(e) \in bd(y) \wedge x \neq y$,
- $lab : V \cup B \cup E \rightarrow \Sigma$ is a labelling function,
- $atr : V \cup B \cup E \rightarrow 2^A$ is an attributing function.

It should be noted that between the same fragments of two components there can exist more than one relation. In this case, between bonds corresponding to these fragments several edges are created. However, for reasons of clarity, we draw only one edge and attach to it a set of labels representing all relations that take place between the bonds.

Example 2. A CP-graph representing the layout of the designed swimming pool building is presented in Fig. 5. This CP-graph is composed of thirteen nodes with bonds assigned to them drawn as small circles. Nodes represent rooms, while their bonds represent room walls. There are fourteen edges that represent the accessibility relation (denoted *acc*), and another fourteen representing the adjacency relation (denoted *adj*). There are also seven edges that represent the visibility relation (the label *vis* is attached to seven edges representing also the accessibility or the adjacency relations).

The selected parts of a created design and requirements specified using dialog windows also have their internal representations in the form of CP-graphs. CP-graphs representing the visibility relation between the restaurant and the main swimming pool (Fig. 4), the floor plan composed of the swimming area in the south, communication area in the east and changing area in the north, and the swimming area composed of a main pool and sauna (Fig. 3) are shown in Fig. 6.

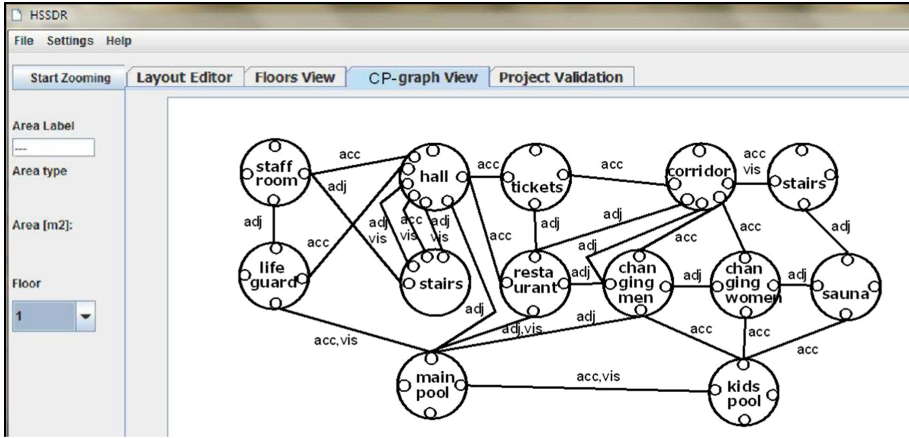


Fig. 5. A CP-graph representing the floor layout of the swimming pool from Fig. 4.

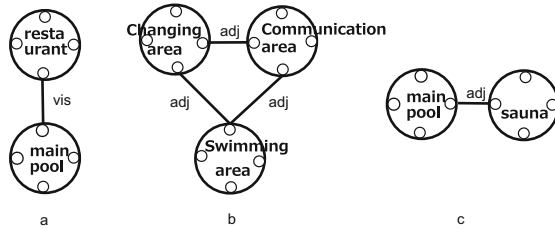


Fig. 6. CP-graphs representing design requirements.

4 CP-Graph Transformation Rule Inference

When structures of designed objects are described in terms of CP-graphs, they can be effectively generated and modified using CP-graph transformation systems. A CP-graph transformation system is composed of a set of transformation rules, which can be applied to object representations in the form of CP-graphs as long as no more of them are applicable. Based on design actions taken by the designer during the design process, CP-graph transformation rules are constructed. As successive sequences of design actions performed by the designer can be seen as steps that develop the generated solution, they correspond to various CP-graph transformation rules, which develop the corresponding CP-graph representation of the designed object. Dividing one area into several spaces in the process of designing the floor layout results in generating a rule with the left-hand side being a node labelled by the name of the divided area and the right-hand side being a CP-graph composed of nodes representing new spaces and edges representing relations between these spaces. Additional graph transformation rules are created based on design patterns and requirements specified by the designer. The obtained set of CP-graph transformation rules is used to

automatically generate CP-graphs corresponding to new objects, some of which can be unexpected and innovative.

In order to apply graph transformation rules some bonds of CP-graphs occurring in these rules are specified as external ones. Let $G = (V, B, E, bd, s, t, lab, atr)$ be a CP-graph. A partial function $ext : B \rightarrow \mathbb{N}$, where \mathbb{N} is a set of integers, determines external bonds of G . A CP-graph transformation rule is of the form $p = (l, r, sr)$, where l and r are attributed CP-graphs with the same number of different values of external bonds and sr is a set of functions specifying the way in which attributes assigned to nodes of l are transferred to the attributes assigned to nodes of r . Each attribute has a set of possible values, which are established during the rule application. The application of the rule p to a CP-graph G consists in substituting r for a CP-graph being an isomorphic image of l in G , replacing external bonds of the CP-graph being removed with the external bonds of r with the same numbers, and specifying values of attributes assigned to elements of r according to functions of sr . After inserting r into a host CP-graph all edges which were coming into (or out of) a bond with a given number in the CP-graph l , are coming into (or out of) bonds of r with the same number.

A CP-graph transformation system GTS is composed of a finite set of rules of the form $p = (l, r, sr)$, where l and r are attributed CP-graphs such that the set of numbers defined for bonds of l by ext is the same as the set of numbers defined by ext for r , i.e., $set(ext(B_l)) = set(ext(B_r))$.

Example 3. Selected CP-graph transformation rules inferred based on design actions taken by the designer during the generation of floor layouts of swimming pools are presented in Fig. 7. Rules $p1$, $p4$, $p6$, and $p8$ are created as the result of dividing the space planned for the layout, to the entrance area, swimming area, changing area, and technical area, and then decomposing the entrance area, swimming area, and changing area into appropriate rooms, in the process of designing the floor layout presented in Fig. 2. Rules $p2$, $p5$, $p7$, and $p10$ are created as the result of dividing the space planned for the layout to the communication area, swimming area, changing area, and technical area, and then decomposing these areas into appropriate rooms, in the process of designing the floor layout presented in Fig. 4. For rule $p2$ the way of specifying values of the attribute *area* assigned to the nodes of the rule right-hand side CP-graph in respect to the value of this attribute defined for the node labelled *Outline* of the rule left-hand side CP-graph is shown. During this rule application, the area of the swimming space is set as half of the whole floor plan area, the changing space is set to 1/6 of the floor area, the communication space and technical space have the areas of 1/4 and 1/12 of the whole floor area, respectively.

Rules $p3$ and $p9$ are created based on the design requirements defined by the designer using dialog windows (Fig. 3) and represented by CP-graphs shown in Fig. 6b and Fig. 6c. The right-hand sides of these rules correspond to composition graphs from Figs. 6b and 6c. Rule $p3$ allows the system to divide the floor plane into the communication area, swimming area, and changing area with establishing values of attributes assigned to the nodes representing these areas

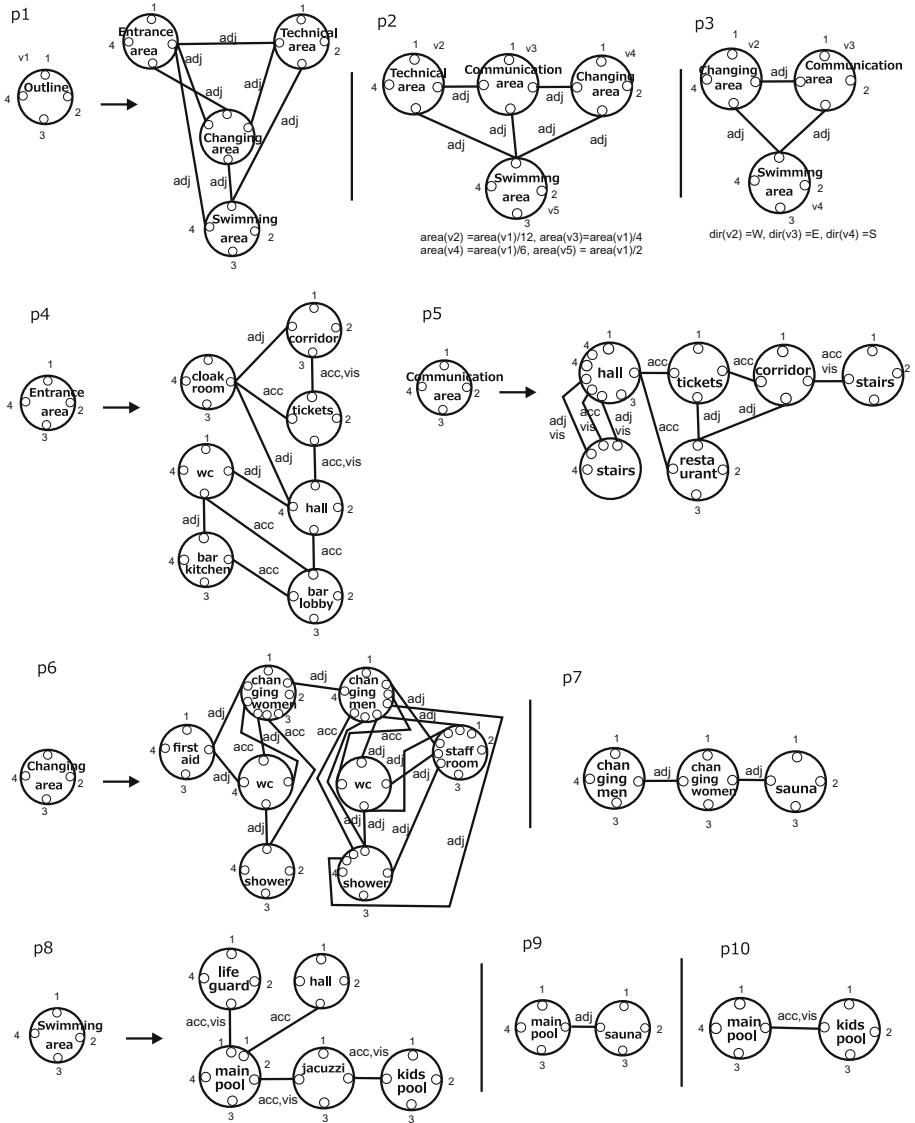


Fig. 7. Selected rules of a CP-graph transformation system inferred based on design actions taken by the designer during generation of floor layouts of swimming pools and specified design requirements

and specifying their world directions to east, south, and west, respectively. Rule *p9* enables the system to generate the swimming area composed of a main pool and sauna.

The CP-graph shown in Fig. 6a, which represents the requirement that the main pool should be visible from the restaurant, leads to creating two rules

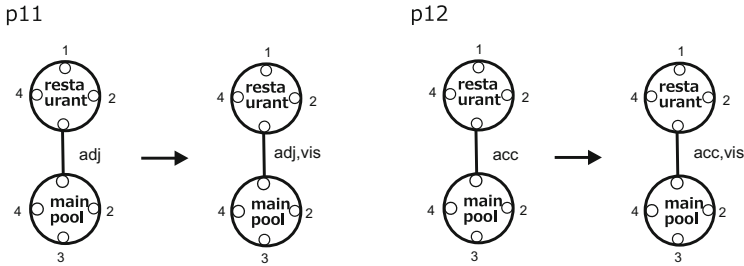


Fig. 8. Two CP-graph rules which ensure the visibility relation between the restaurant and the main pool.

presented in Fig. 8. Rule *p11* enables the system to add the visibility relation between the restaurant and the main pool, which are adjacent to each other, while rule *p12* enables to add the visibility relation between the restaurant and the main pool, which are accessible from each other.

A CP-graph transformation system, which selected rules are presented in Figs. 7 and 8, generates CP-graph representations of floor layouts of swimming pools. The embedding transformation for all these rules is such that all edges which are connected in the host CP-graph with the bond number *i* of the left-hand side of a production are replaced by edges connected with all bonds with number *i* on the right-hand side of this rule.

5 Automatic Generation of New Deigns

Combining the application of transformation rules inferred from the generation process of various designs in one derivation allows the system to generate CP-graphs representing new and sometimes innovative designs. These solutions can be treated as an inspiration for the designer, who can slightly modify them or develop further. At first, rules inferred based on the designer actions, are applied as long as it is possible starting from the CP-graph representing the initial design. Then to the obtained CP-graph, the rules changing design patterns or maintaining design requirements can be applied as long as the designer wants to or no more rules are applicable.

Example 4. Two floor layouts of the swimming pools, which correspond to CP-graphs generated using CP-graph transformation rules inferred based on design actions taken by the designer during generating swimming pool floor layouts (see: Figs. 7, 8, 10) are shown in Fig. 9. In the first case, rule *p13* which locates a swimming area at the south and a communication area at the north (Fig. 10) was used. Then the swimming area was replaced by a main pool using rule *p18* shown in Fig. 10. In the next step rule *p7* from Fig. 7 was used to divide the changing area into three rooms, and at the end rule *p15* (Fig. 10) was applied to divide the communication area into a corridor, ticket office, and a hall. In the second case the rules *p14*, *p18*, *p17*, and *p16* were used.

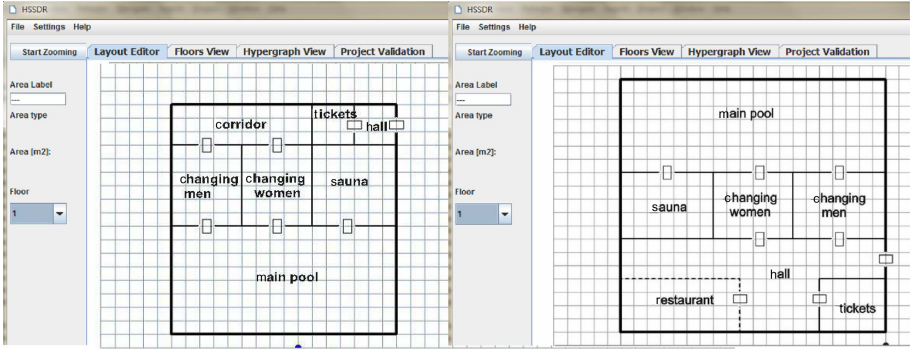


Fig. 9. Two floor layouts of swimming pools.

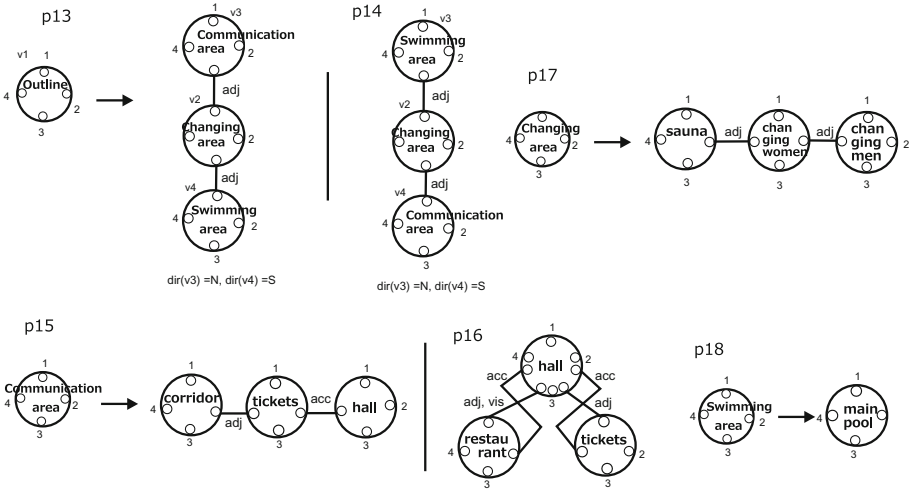


Fig. 10. The other CP-graph transformation rules inferred based on design actions taken by the designer during the generation of floor layouts of swimming pools.

6 Conclusion

In this paper, the problem of supporting the conceptual phase of the design process by inferring CP-graph transformation rules from initial designs, specified requirements, and design patterns has been considered. Initial designs are internally represented by graph structures, which are obtained automatically during the design process and can be transformed using graph rules. Based on design actions taken by the designer during the design process, CP-graph transformation rules are constructed. Additional CP-graph transformation rules are created on the ground of design patterns and requirements specified by the designer.

The obtained set of CP-graph transformation rules is used to automatically generate CP-graphs corresponding to new objects, some of which can be unexpected and innovative, and enables the designer to experiment with his/her own design ideas.

In future work, the memory of CP-graph rules will be used to ensure the presence of a predefined number of selected components within a designed object and to preserve some required characteristics of the design. We will also investigate such internal mechanisms of our design system as an ability to learn and a mechanism that would enable automatic evaluation of the obtained solutions.

References

1. Borkowski, A., Grabska, E.: Representing designs by composite graphs. In: IABSE Colloquium on Knowledge Support Systems in Civil Engineering, IABSE Reports 72, Bergamo, pp. 27–36 (1995)
2. Chung, C., Wang, S.: Computational Analysis and Design of Bridge Structures. CRC Press, Boca Raton (2014)
3. Goel, A.K., Rugaber, S.: Interactive meta-reasoning: towards a CAD-like environment for designing game-playing agents. In: Besold, T.R., Schorlemmer, M., Smaill, A. (eds.) Computational Creativity Research: Towards Creative Machines. ATM, vol. 7, pp. 347–370. Atlantis Press, Paris (2015). <https://doi.org/10.2991/978-94-6239-085-0-17>
4. Grabska, E.: Graphs and designing. In: Schneider, H.J., Ehrig, H. (eds.) Graph Transformations in Computer Science. LNCS, vol. 776, pp. 188–202. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-57787-4_12
5. Grabska, E., Ślusarczyk, G., Gajek, S.: Knowledge representation for human-computer interaction in a system supporting conceptual design. *Fundamenta Informaticae* **124**, 91–110 (2013)
6. Grabska, E., Strug, B., Ślusarczyk, G.: Cooperative design in a visual interactive environment. In: Luo, Y. (ed.) CDVE 2018. LNCS, vol. 11151, pp. 153–162. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00560-3_21
7. Grabska, E., Strug, B., Ślusarczyk, G.: A visual interactive environment for engineering knowledge modelling. In: Smith, I., Domer, B. (eds.) Advanced Computing Strategies for Engineering. EG-ICE 2018, LNCS, vol. 10863, pp. 219–230. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91635-4_12
8. Mars, A., Grabska, E., Ślusarczyk, G., Strug, B.: Design characteristics and aesthetics in evolutionary design of architectural forms directed by fuzzy evaluation. *AI EDAM* **34**, 147–159 (2020)
9. Pahl, G., Beitz, W., Feldhusen, J., Grote, K.-H.: Engineering Design: A Systematic Approach. Springer, London (2007). <https://doi.org/10.1007/978-1-84628-319-2>
10. Pipinato, A. (ed.): Innovative Bridge Design Handbook, 1st edn. Butterworth-Heinemann, Elsevier, Amsterdam (2015)
11. Ryszka, I., Grabska, E.: GraphTool - a new system of graph generation. In: Omatu, S. (ed.) ADVCOMP 2013, Porto, pp. 79–83. Curran Associates (2013)