Knowledge Representation in Visual Design

Ewa Grabska, Grażyna Ślusarczyk, Szymon Gajek
The Faculty of Physics, Astronomy and Applied Computer Science,
Jagiellonian University,
Reymonta 4, 30-059 Kraków, Poland
ewa.grabska@uj.edu.pl, gslusarc@uj.edu.pl, szymon.gajek@gmail.com

Abstract—**This paper deals with an approach to represent design knowledge and reasoning in computer systems supporting visual design. Both forms and layouts functionality of designed artifacts are visualized as design drawings with the use of a visual editor. Design knowledge about these drawings is formally represented in the framework of a computational ontology for design. This ontology allows the system to convert drawings into their internal representations. The defined ontological commitment transforms design knowledge encoded in the internal representations of drawings into logic formulas. The obtained logic language enables the system to reason about compatibility of designs with specified constraints. The presented approach is illustrated on the example of designing an indoor swimming pool.**

Keywords-design knowledge; CAD system; conceptual design; visual language; hypergraph

I. INTRODUCTION

This paper aims at introducing a new approach to represent design knowledge and knowledge-based reasoning in computer systems supporting the conceptual phase of visual design. The conceptual design takes place at various levels of abstraction. To externalize the design concepts and ideas the designer usually sketches in search of shapes and relations among them. Contemporary, the designer has the possibility to use specialized CAD tools (like ArchiCAD, Allplan, Revit [1-3]) in order to replace sketches by drawings.

In this paper, we present the knowledge representation, where the visualizations of early solutions made by the designer are the main source of knowledge about created designs. The structure of the design system consistent with the proposed method is presented in Figure 1. Based on this structure the prototype computer-aided visual design system is implemented. It serves to test designing of both two-dimensional floor-layouts and three-dimensional forms of buildings.

In the prototype system the designer creates solutions of a design task using a design interface composed of a visual editor and a rule editor. The former allows the designer both to describe the form and the functionality of the design artifact by means of problem-oriented visual languages. The latter enables him to specify design constraints which should be obeyed during the whole design process.

Designs with required functionality created in this system are represented as drawings forming specified visual languages. These drawings have their internal representations which encode the design knowledge about the design task solutions. Design knowledge is formally represented in the framework of computational ontologies [4]. An ontology for design is defined using a notion of a conceptualization which specifies concepts that are assumed to exist in a given design domain and relationships that hold among them [5]. In our approach the conceptualization is related to the internal representation of a drawing in the form of a specific graph (hypergraph), where graph atoms represent concepts and relations.

To make the design knowledge computer readable we use a first-order logical language to express it in a formal way. We specify the mapping called ontological commitment between elements of the vocabulary of this language and entities of the conceptualization. This mapping allows us to translate design knowledge captured in graph structures into logic formulas describing design drawings. The obtained logic language enables the system to reason about compatibility of designs with specified constraints.

The presented approach will be illustrated by the running example of designing an indoor swimming pool taking into consideration both its form and floor-layout.

The paper is organized as follows. Section 2 describes the related work. In Section 3, visual languages used in the visual design process are described. In Section 4 graph-based internal representations of design drawings are presented. Reasoning based on design knowledge encoded in the internal representation of solutions and translated into logic formulas is discussed in Section 5. The paper ends with a conclusion.

II. RELATED WORK

To construct knowledge-based design systems the representation and manipulation of knowledge in computers is needed [6]. Knowledge-based design systems, in which knowledge pertaining to a given design domain is represented, are integrated with CAD tools to facilitate design process [7-11]. Contemporary CAD systems are expected to extend their functionality far over merely producing drawings. These systems, following the Building Information Modeling paradigm [12] store all project's 3D elements in a central database and are able to generate 2D

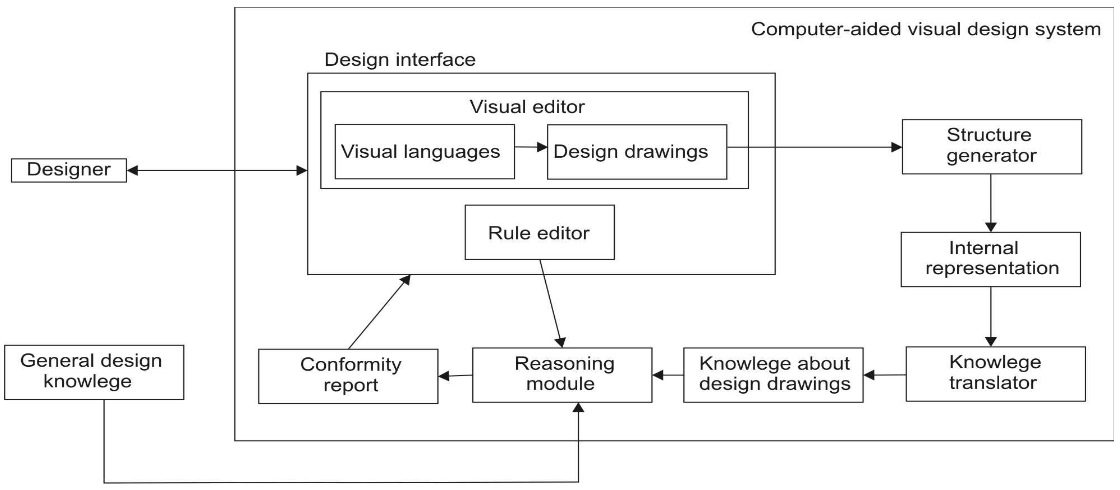

Figure 1. The schema of the computer-aided visual design system

drawings and 3D renderings. However, the most of these tools do not provide data structures related to designs and reflecting the design knowledge extracted from drawings being visualizations of designer ideas. In this paper it is shown how design drawings created by means of the visual editor can be automatically transformed into their internal computer representations.

There are two types of knowledge representations: symbolic and graphical ones. In the former, knowledge is represented explicitly in symbolic terms and reasoning is the manipulation of these terms [13, 14]. In the latter, the way of organization, processing and manipulation of knowledge is based on the spatial relations between objects [15-17]. In this paper both types of knowledge representations are used in such a way that the symbolic representation is based on the graphical one. We propose an extension of the approach presented in [15, 16, 18] where a visual representation of designs made by the designer is created simultaneously together with their internal representations. In this method initial visualizations of designer's solutions are the main sources of knowledge about designs. Up to now this approach has been used to functional designing two-dimensional floor-layouts. At present the designs of architectural forms are also studied based on this method. Design knowledge can be formally represented in the framework of computational ontologies [4]. The ontological framework used in this paper facilitates the description of the proposed design cycle.

Graphs and hypergraphs are used quite frequently in knowledge-based design tools [19]. Our approach is based on a formal model of hypergraphs introduced in [20] and extended in [21].

III. VISUAL LANGUAGES

In this section, we present visualizations of design solutions, which can be created by the designer in different phases of the visual design process.

On the basis of general requirements concerning a design task the designer starts with creating a three-dimensional visualization being a general form of an artifact. Then an outline of the floor is created as the intersection of this form with a plane at a given height. Such a shape can be also treated as a starting point of the design process when a floor-layout is created on the basis of functional aspects of the solution. At the outset of this process the conceptualization is modeled, i.e., the relevant entities and relations emerging from the design task under consideration are specified. In other words, having a 2D contour of a design object the designer describes the inner structure of the object. In our approach this structure is obtained by means of a visual editor and it is a floor-layout visualized as a design drawing.

The designer communicates with the design system using a visual editor which enables him to use different visual design languages. A visual language is a set of design drawings being configurations of basic shapes. Thus it is characterized by a vocabulary being a finite set of basic shapes and a finite set of rules specifying possible configurations of these shapes. The basic shapes of visual languages and their spatial relationships correspond to concepts and relations defined by the conceptualization of the design domain. During a design process each visual language allows the designer to specify the specialization hierarchy of design concepts.

Example. Let us consider an example of designing an indoor swimming pool in a one-storey building. At first a form of this building is created by the designer with the use of a 3D visual language (Figure 2). The vocabulary of the used language is composed of cubes which can be translated, rotated and scaled or undergo Boolean operations. It is worth noticing that the existing graphical tools like Revit [3] or ICE [22] can be used in this design phase. The 2D contour of the form located in an orthogonal grid is shown in Figure 3a. In the next step, taking into consideration the type of the swimming pool (recreational, sports, learner) and an approximate number of users, the designer decomposes this contour into functional areas represented by polygons

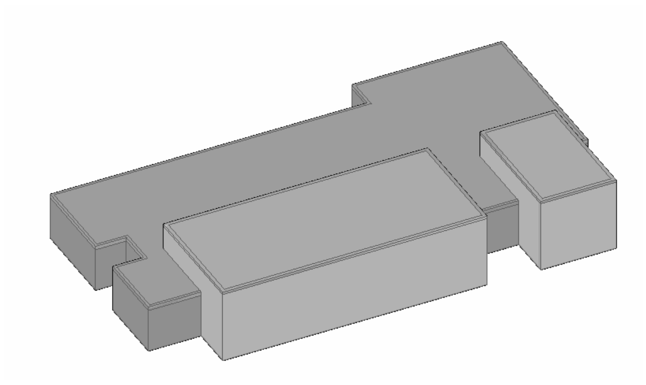
Figure 2. The proposed form of a swimming pool building

(Figure 3b). 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 cloak-room, 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. The obtained whole floor layout is shown in Figure 4. The vocabulary of the language, which enables the designer to design floor layouts, is composed of shapes corresponding to components like rooms, walls, and additional graphical symbols allowing the designer to express the relations between components. In Figure 4, segments with dashed lines represent the visibility relations among components, continuous segments shared by polygons denote the adjacency relations between them, while segments with rectangles on them represent the accessibility relations.

IV. THE INTERNAL REPRESENTATION OF DESIGN DRAWINGS

Our approach deals with visual designing i.e., during the design process the designer communicates with the system by means of design drawings automatically transformed by the structure generator module into their internal representations in the form of graphs. The graph structure enables the system to store the knowledge about syntactic aspects of created drawings [23]. Graph atoms represent concepts and relations corresponding to the elements of the conceptualization determined by the designer. To represent the top-down way of designing as well as the hierarchy of design concepts, a hierarchical data structure is needed [20, 21].

The internal data structure used in our approach is called an attributed hierarchical hypergraph. The prefix ‘hyper’ in the word ‘hypergraph’ denotes that this graph structure allows for expressing multi-argument relations between drawing components. The considered hypergraphs are composed of object hyperedges corresponding to layout components and relational hyperedges, which represent relations among fragments of components. The fragments of components that can be used as arguments of relations are represented by hypergraph nodes. Hyperedges are labelled by names of components or relations.

Drawing a design diagram the designer specifies labels of components related to room types. While he creates the diagram and/or modifies it using design actions, the hierarchical hypergraph is automatically generated. In our algorithm, for each labeled design component in the form of a polygon, one object hyperedge is created. Semantic information about this component describing it as a room is automatically completed by a hyperedge label describing a type of this room. When the designer divides a component into parts, the hierarchical hypergraph composed of object and relational hyperedges representing the arrangement of these parts is nested in the object hyperedge representing the divided component. For each line shared by polygons in the diagram one relational hyperedge connecting nodes representing corresponding sides of the polygons is generated. Semantic information about this relation depends on the line style and determines the type of the relational hyperedge label. The continuous lines correspond to adjacency relations, the dashed lines represent visibility relations, while the lines with small rectangles on them correspond to accessibility relations.

To represent design features being other type of semantic information concerning layout components, attribution of nodes and hyperedges is used. Attributes represent properties (like shape, size, position, material) of elements corresponding to object hyperedges and nodes.

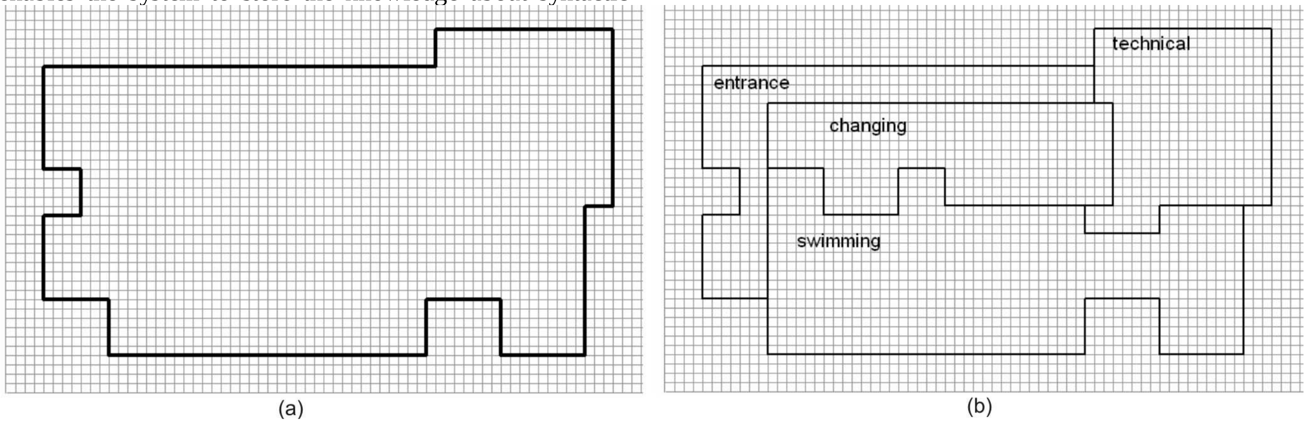

Figure 3. (a) The contour of the swimming pool, (b) the decomposition of the contour into four functional areas

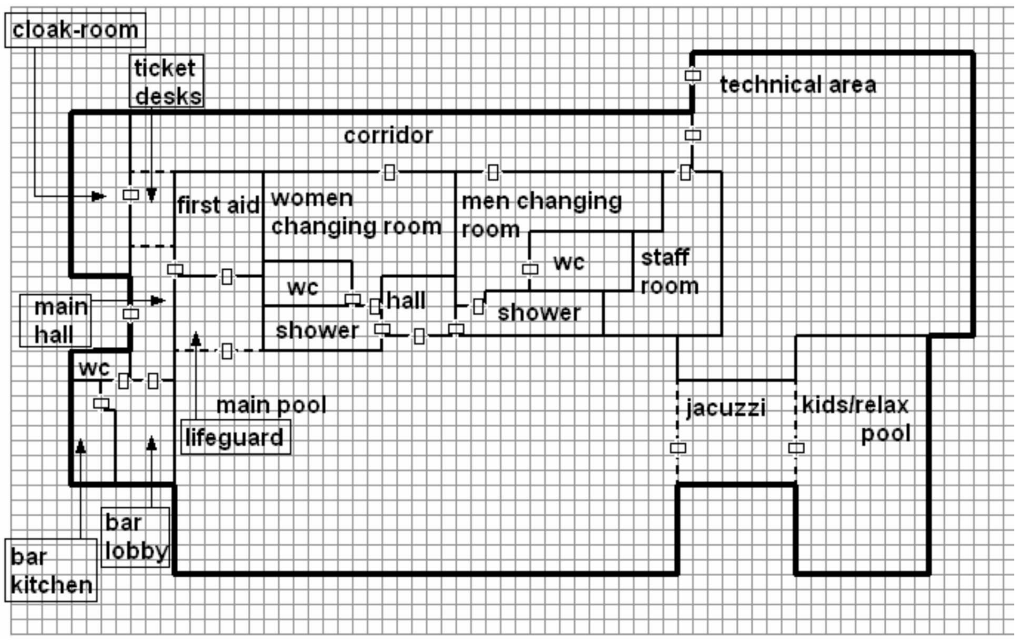

Figure 4. The floor layout of the swimming pool

The values of such attributes as *area* are automatically set by the system at the time of creating rooms on the basis of the occupied part of the grid on which diagrams are drawn. The values of other attributes, like *material* characterizing walls, can be specified by the designer at the time of establishing the relations between rooms.

Example. A fragment of the hierarchical hypergraph representing the layout of the designed swimming pool is presented in Figure 5. This hypergraph is composed of fourteen object hyperedges (denoted by rectangles) and thirty three non-directed relational hyperedges (denoted by ovals), where nine of them represent the accessibility relation, twenty three represent the adjacency relation and one represents the visibility relation. The swimming area and the changing area are represented by hierarchical object hyperedges. The nested object hyperedges correspond to the rooms in these areas. Hypergraph nodes represent walls of areas or rooms and are assigned as target nodes to object hyperedges representing these areas or rooms. The values of the attribute order are shown near nodes. They determine the order of walls as well as their level of hierarchy. The first element of the sequence denotes the number of a wall corresponding to one of the polygon sides, while the length of this sequence determines in which design step the wall was introduced.

V. REASONING BASED ON DESIGN KNOWLEDGE

In order to express the design knowledge about generated drawings in a formal way a first-order logical language is used [13]. We specify the mapping called ontological commitment between elements of the vocabulary of this language and entities of the conceptualization. The layout components are assigned to the constant symbols, their attributes are assigned to function symbols, while relations between the components correspond to the predicate symbols. This commitment allows the system to transform semantic and syntactic information encoded in the internal representations of drawings into logic formulas.

In the running example, elements of a set of concepts (areas, rooms, walls) and relations (adjacency, accessibility and visibility) of a given visual language are associated with symbols of the vocabulary of the logic language. The sides of polygons representing walls of rooms are associated with constant symbols, while lines shared by polygons representing relations between them are assigned to predicate symbols. Attributes determined for walls and rooms correspond to function symbols.

The formulas are interpreted using a relational structure, which assigns hypergraph nodes, object hyperedges and their attributes to the terms, and relational hyperedges to the predicates of the formulas. The logical language stores knowledge about created designs and enables the system to reason about compatibility of designs with specified constraints. It is worth noticing that design knowledge can be also expressed in different types of logic (e.g., propositional logic), depending on the considered design problem.

The system is also equipped with a knowledge base composed of formulas expressing general design knowledge specific to a particular design task. Restrictions and rules of this base describe design standards like architectural norms, fire regulations, etc. Additionally, there exists the possibility to specify designer's own requirements and restrictions using a rule editor being a part of a design interface.

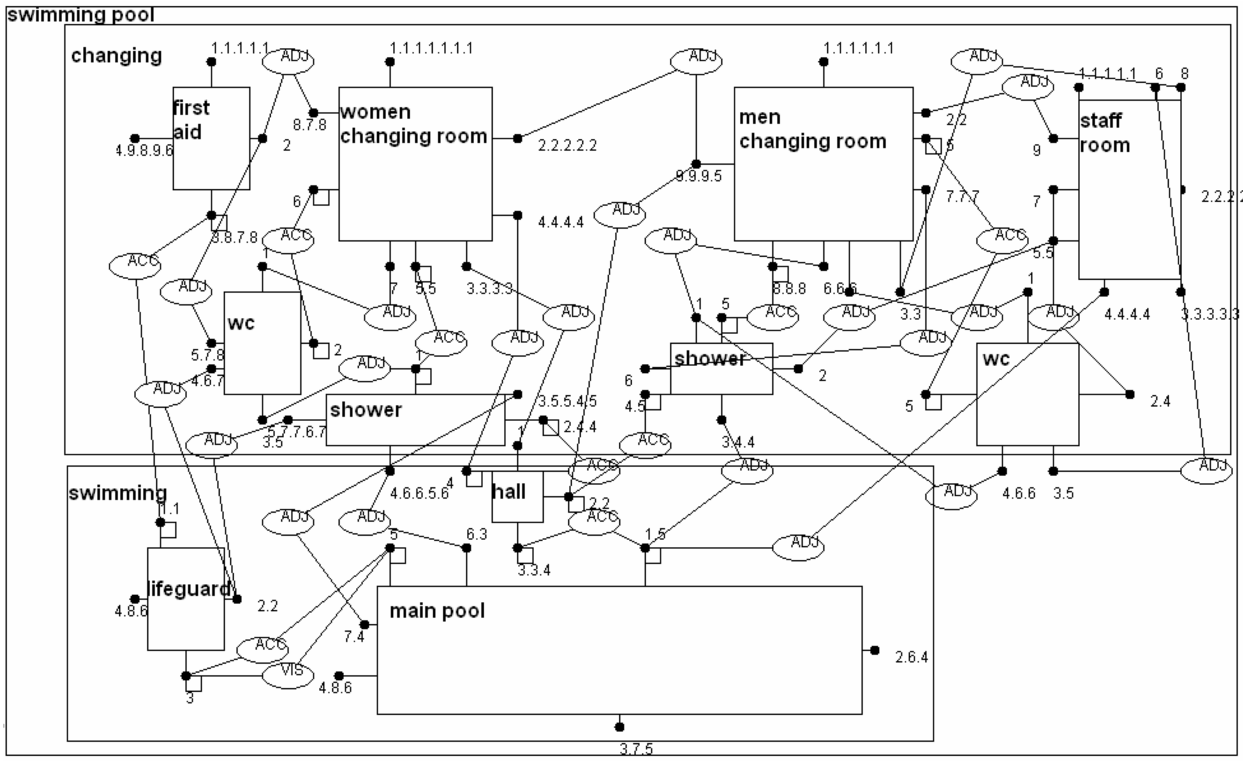

Figure 5. A fragment of the hierarchical hypergraph representing the layout of the swimming pool

All three kinds of formulas, namely formulas describing the created designs, formulas describing general design knowledge and rules defined by the designer, enable the system to support the user in creating admissible, acceptable and safe artifacts [24]. The reasoning module of the system checks the conformity of the design drawing representing the created design task solution with the specified design criteria. Then the conformity report is presented to the designer through the design interface.

Example. The knowledge stored in the hierarchical hypergraph presented in Figure 5 is translated into propositional logic formulas describing relations between rooms of the designed layout (Figure 4), like *visibility* (*lifeguard_room*, *main_pool*), *adjacency*(*staff_room*, *main_pool*), *accessibility*(*corridor*, *women_changing_room*). For instance the *visibility* relation between two rooms holds if there exist two hypergraph nodes representing walls and assigned to two different component hyperedges (representing rooms) and to the same relational hyperedge labeled *visibility*. Moreover the attributes assigned to these nodes and specifying the wall material should have the value corresponding to the glass.

For the drawn layout the system automatically calculates the values of the attribute specifying the area of rooms (e.g., *area*(*main_pool*) = 2120). Then the reasoning module can check the agreement between the proposed layout and standard architectural norms for swimming pool designs [25-27]. For example it test if the conditions *area*(*women_changing_room*) ≥ *area*(*main_pool*)/7, and *area*(*main_pool*)/20 ≤ *area*(*shower*)+*area*(*wc*) ≤ *area*(*women_changing_room*) are satisfied. Checking the fire regulations the system computes also the distance from all rooms to the main hall.

In the next step the conformity of the solution with constraints defined by the designer is checked. For example the system can check whether the rectangular swimming pool is placed with its longer wall towards the South. This constraint is satisfied if the following condition holds: *if length*(*main_pool*.1) > *length*(*main_pool*.2) *then location*(*main_pool*.1) = *S or location*(*main_pool*.1) = *N else location*(*main_pool*.2) = *S or location*(*main_pool*.2) = *N*, where *main_pool*.1 and *main_pool*.2 correspond to nodes, which are assigned to the hyperedge representing the main pool and denote the walls of the pool (in this case nodes with numbers 4.8.8.5 and 3.7.7.4) , while *length* and *location* are attributes specifying the length and geographical orientations of walls.

The hierarchical representation of design knowledge facilitates the reasoning process. Let us consider the formula which allows one to check "if there exists a staff room with the area of at least 10m^2 and located in the changing area". The formula is as follows: $\exists x, y$: $lb(y)$ = *changing* $\wedge$ $x \in ch^{+}(y) \wedge lb(x)$ = *staff_room* $\wedge$ $area(x) \geq 10$, where x and y are variables, lb is a hypergraph labeling function, ch^{+} is a function determining all ancestors of the given hyperedge.

After the design of the 2D floor layout is completed, the general form of the building with internal walls dividing the rooms is visualized (Figure 6). This visualization gives the designer the possibility to simulate the object behavior.

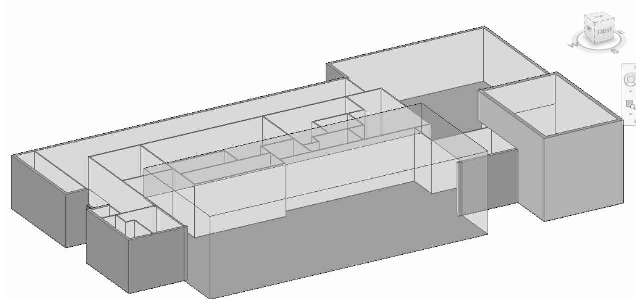
Figure 6. The form of the building with internal walls dividing the rooms

VI. CONCLUSION AND FUTURE WORKS

In this paper, an approach to visual design of buildings was considered. Different visual languages are used to design architectural forms and floor-layouts of buildings. A rule editor of the presented system enables the designer to specify design constraints which should be obeyed during the whole design process. The design drawings obtained during the design process have their internal representations in the form of attributed hierarchical hypergraphs encoding the design knowledge about the drawings. This knowledge is translated into logic formulas. The obtained logic language enables the system to reason about compatibility of designs with specified constraints by comparing its formulas with formulas expressing design knowledge specific to particular design tasks and requirements defined by the designer.

In the future work the multi-storey buildings will be designed with the use of visual languages. It will require extending generation methods used to create architectural forms to include for instance 3-D shape grammars which are present subjects of our studies.

REFERENCES

[1] ArchiCAD, Graphisoft, http://www.graphisoft.com/products/archicad/, 15.07.2011.

[2] Allplan Architecture, Nemetschek, http://www.allplan.com/, 15.07.2011.

[3] Autodesk Revit Architecture, Autodesk, http://www.autodesk.com/revit, 15.07.2011.

[4] N. Guarino, D. Oberle, and S. Staab, "What is an Ontology?", Handbook on Ontologies, S. Staab, R. Studer, Eds. International Handbooks on Information Systems, Springer-Verlag, 2009.

[5] M.R. Genesereth and N.I. Nillson, Logical Foundations of Artificial Intelligence, Morgan Kaufmann, Los Altos, CA, 1987.

[6] R.D. Coyne, M.A. Rosenman, A.D. Radeford, M. Balachandran, and J.S. Gero, Knowledge based design systems, Addison-Wesley Publishing Company, 1990.

[7] A. Goel, S. Bhatta, and E. Stroulia, "KRITIK: An early case-based design system" in M.L. Maher and P. Perl, Eds., Issues and Applications of Case-Based Design, Erlbaum, Hillsdale, NJ, 1997, pp.87-132.

[8] U. Flemming and R. Woodbury, "Software Environment to support the Early phases in building Design (SEED): Overview, J. Arch. Engrg. 1, 147, 1995.

[9] U. Flemming, R. Coyne, T. Gavin, and M. Rychter, "A generative expert system for the design of building layouts - version 2", in B. Topping, Ed., Artificial Intelligence in Engineering Design, Computational Mechanics Publications, Southampton, 1999, pp.445-464.

[10] K. Sycara, R. Guttal, J. Koning, S. Narasimhan, and D. Navinchandra, "CADET: a Case-based Synthesis Tool for Engineering Design", International Journal of Expert Systems 4, 1992.

[11] A. Voß, "Case design specialists in FABEL", in M.L. Maher and P. Perl, (Eds.), Issues and Applications of Case-Based Design, Erlbaum, Hillsdale, NJ, 1997, pp.301-336.

[12] Ch. Eastman, P. Teicholz, R. Sacks, and K. Liston, BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors, Wiley, 2008.

[13] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi, Reasoning About Knowledge, MIT Press, Cambridge, 1995.

[14] S.Garavaglia, PROLOG, Harper and Row, New York, 1984.

[15] E. Grabska, G. Ślusarczyk, and T.L. Le, "Visual design and reasoning with the use of hypergraph transformations", Proc. of the Seventh International Workshop GT-VMT 2008, Budapest, Electronic Communications of EASST 10, 2008, pp. 305-318.

[16] E. Grabska, A. Łachwa, G. Ślusarczyk, K. Grzesiak-Kopeć, and J. Lembas, "Hierarchical layout hypergraph operations and diagrammatic reasoning", Machine GRAPHICS and VISION, vol. 16, 2007, pp. 23-38.

[17] M. Nagl, (Ed.), "Building Thightly-Integrated (Software) Development Environments: The IPSEN Approach", LNCS 1170, Berlin: Springer Verlag, 1996, pp.248-279.

[18] E. Grabska, A. Borkowski, W. Palacz, and S. Gajek, "Hypergraph system supporting design and reasoning", EG-ICE International Workshop, Berlin, Germany, 2009.

[19] A. Schurr, A. Winter, and A. Zundorf, "Graph grammar engineering with PROGRES", in W. Schafer and P. Botella Eds., Proc. of the 5th European Software Engineering Conference (ESEC95), LNCS 989, Springer-Verlag, Berlin, 1995, pp. 219–234.

[20] F. Drewes, B. Hoffmann, and D. Plump, "Hierarchical Graph Transformation", Proc. of FOSSACS 2000, J. Turyn, Ed., LNCS, vol. 1784, Springer, 2000, pp. 98–113.

[21] G. Slusarczyk, "Hierarchical Hypergraph Transformations in Engineering Design", Journal of Applied Computer Science, 11(2), 2003, pp.67–82.

[22] O. Akin and H. Moustapha, "Formalizing generation and transformation in design", Proc. Design Computing and Cognition'04, J.S. Gero, Ed., Kluwer, Dordrecht, 2004, pp. 176-196.

[23] B. Kraft, "Conceptual Design Tools for Civil Engineering", Proc. of AGTIVE'03, 2003, pp. 434-439.

[24] E. Grabska and G. Ślusarczyk, "Knowledge and reasoning in design systems", Autom. Constr., in press, doi:10.1016/j.autcon.2011.03.009.

[25] D. Fabian, Bäderbauten: Aquatic buildings (Handbuch für Bäderbau und Badewesen: Anlage, Ausstattung, Betrieb, Wirtschaftlichkeit), Verl. Georg D. W. Callwey, München, 1970.

[26] E. Neufert, Architects' Data, BSP Professional Books, Oxford, 1990.

[27] R. Wirszyłło, Ed., Sport equipment. Designing and building (in Polish), Arkady, Warszawa, 1966.