Bridging the Gap in Constraint-Based Design

VELIZ Alejandro, MEDJDOUB Benachir and KOCATURK Tuba
Design Directorate, School of the Built Environment, University of Salford, United Kingdom
a.a.veliz@edu.salford.ac.uk, b.medjdoub@salford.ac.uk, t.kocaturk@salford.ac.uk

Abstract. Mass customization is one of the most promising computational developments in the AEC industry. Despite recent advances in the production of research-based knowledge, the professional practices lack of a consistent and permanent technology adoption scheme and remain as a very resilient and fragmented industry. This work is a part of an ongoing research project developing guidelines for improving both physical and virtual modeling processes within an architectural design context. Here, we present a customizable model of a space layout explorer. The implementation of the user-driven solution-finding process is based on constraint technology embedded in Autodesk's Revit® 2011 macros tools, commonly used in the professional practice. The aim of this work is to demonstrate a practical use of a small constraint-based system on software of widespread use. Even though there is still a lack of building information, the model has already several applications in the definition a floor plan layout and in the comparison of several instances of the design solution in the 3D user view. User-driven modifications are not made directly through the 3D model, but through different explicit text tags that describe each parameter on 2D views - although a real time 3D visualization of the model is also available-. The main findings are discussed as guidelines for further research on the end-user involvement on a "creative mass customization" scheme. Also, the implementation of visual aids such as text tags during the customization process can bridge some technical obstacles for the development of interfaces for constraint-based mass customization systems. Before the final discussion, some limitations on the use of this model are described.

CAAD Futures 2011 : Designing Together, ULg, 2011

1. Introduction

Despite the massiveness of some digital tools and methods during the architectural design process such as computer-aided drawing, 3d modelling and simulation analysis, it is also necessary to explore how to bridge the gap between a trend of emerging research-based knowledge and a very resilient industry : in 1956 Gropius estimated a technology adoption span of 25 years [1], and more recently Larson [2] has quantified in 17 years the time that innovations take to find their way into the housing industry. The cause for the reality gap in CAAD [3, 4] has been described among other reasons, as a lack of collaboration between researchers and practitioners, and the failure of CAAD softwares in providing a full view of the design phenomenology. This gap has caused a non-standardized and low-innovative market [2], and a fragmented construction industry with difficulties to share and reuse knowledge [5].

Duarte [6] affirms that the quality of a house is directly related with the satisfaction of the end-user needs. So far, most of the times end-user needs have been translated into design information by using heterogeneous methods such as small scale models, color diagrams, mapping, polls, and the final solution is finally a designer's interpretation on a try to accomplish as most requirements, as close as possible. In contrast with this, the 'creative mass customization' scheme [7] entails the user participation not just in the final stages of the productive chain, but also on the design, fabrication and assembly processes; end-user needs cause changes on the very early stage of the design process itself. As a response to this theoretical foundation, the concept of "One Size fits None" was arosed in [8] as a possible graphic-user interface that allows the implementation of mass customization tools in architectural design.

When it comes to constraint-based design, a wide set of approaches have arisen from research, from constraint-based modeling and solution-finding engines, i.e. [9-12] to constraint-based checking of existing models in compliance to design regulations [13]. According to the roles for explicit knowledge described in [14], the use of research-based knowledge in organizations entails solving different types of problems, or connecting people with valuable or reusable knowledge. In this work, we introduce a first prototype of an ongoing research that will develop guidelines for improving both physical and virtual modeling processes within an architectural design context. There is evidence that a constraint-based method is not only linked with automated solution-finding design tools, but also allows human-computer interactions by involving end-users on the quest for solutions of specific design problems [11]. We review and describe used tools in constraint-based architectural design engines in Section 2. Implementation of a floor plan layout model made in Autodesk's Revit 2011 is presented in Section 3. Before the discussion, the exploratory model is presented and its limitations are described in Section 4.

2. Bridging the gap

Constraint-based design engines have been developed in several different softwares. Whilst some authors program their own engines, others take advantage of existing softwares' embedded tools. Table 1 describes some tools and constraint-based systems previously used in architectural design-related research.

Table 1. Tools and constraint-based systems created to assist different architectural design tasks and processes.

REF.	LANGUAGE / IMPLEMENTATION	APPLICATION
[9]	Jsolver - ECLiPSe / Microstation 3D	Floorplan layout design problem
[10]	Jsolver - ECLiPSe / Microstation 3D	Plant room design problem
[11]	OPL / OPL Studio	Bulk design problem
[11]	Xpresso / Cinema4D	Bulk Design Problem
[12]	Xpresso / Cinema4D	Sanitary core design problem
[13]	EXPRESS	IFC Checking
[15]	C++	Knowledge-based design method
[16]	EXPRESS	Definition / verification system
[17]	PROLOG	Virtual environment prototype
[18]	PROLOG	Space layout planning
[19]	VRML-Java-HTML	3D design environment
[20]	Linear Programming	Design support system
[21]	Autodesk Revit	Bulk design problem
[22]	Cameleon Model Designer	Customization process assistance
[23]	C	Assembly modeling
[24]	Mac Common Lisp	Construction Kit Builder
[25]	SNOPT Solver	3D modeling from images
[26]	C++	Design of parts
[27]	C++	Geometric constraint solver

Evidently, there is a wide pool of available languages and tools to implement constraint-based design systems and constraint solvers. Despite this, their use in architectural practices is limited due to the lack of text-based programming skills

in the architectural curricula. Some exceptions are made in [11], [12] and [21] where constraint-based systems were implemented on plaftorms that do not require previous knowledge in programming. Specifically, applications described in [11, 12] were built by using visual programming, a method whose implementation is becoming widespread. As an alternative to bridge a general lack of hard programming skills, the following model describes a space layout explorer using macros tools available in Autodesk's Revit 2011. In accordance to Tab. 1, all previous space layout systems have used heavy programming to generate the floor plan solutions.

3. The spatial layout design problem

As a study case, a spatial layout explorer was built, based on a model for a typical apartment previously analyzed by Niemeijer et al [16]. In architectural design and planning, the space layout problems can be solved from several approaches. In this case, a predefined floor plan layout has been used to describe a user-driven variational behaviour for the enclosures of every space. Therefore, the spatial arrangement can be visualized as a result of the positioning and dimensioning of physical enclosures, not as an initial design condition.

The general purpose of this contribution is to demonstrate the use of a simple constraint-based tool in a practical design exercise, guaranteeing that variations can be defined within a framework of fixed and variable conditions :

- The floor plan has a fixed rectangular size of 1160 x 840 cm.
- The model consists of 8 rectangular spaces plus a resulting space of irregular shape (Figure 1). Spaces are therefore understood as areas which are completely enclosed by walls, independently of their shape.
- The sanitary cores (Spaces 2, 3, 7 and 8) have fixed sizes and positions on the floor plan and will not be considered for floor plan variations.
- If the user determines a value outside the solution domain for the dimension in X and Y of any space, that enclosure will disappear. This fact not only proposes an opening of the amount of solutions, but also produces a variable amount of resulting spaces, and changes in the space topology arrangement.
- Every wall has been simplified to a uniform thickness of 10 cm.
- The model does not consider, by now, windows or doors opening operations. However, existing commands in Autodesk's Revit® 2011 allow the designer to easily create voids. The only voids in the model are located in the accesses of the sanitary cores and the main access.

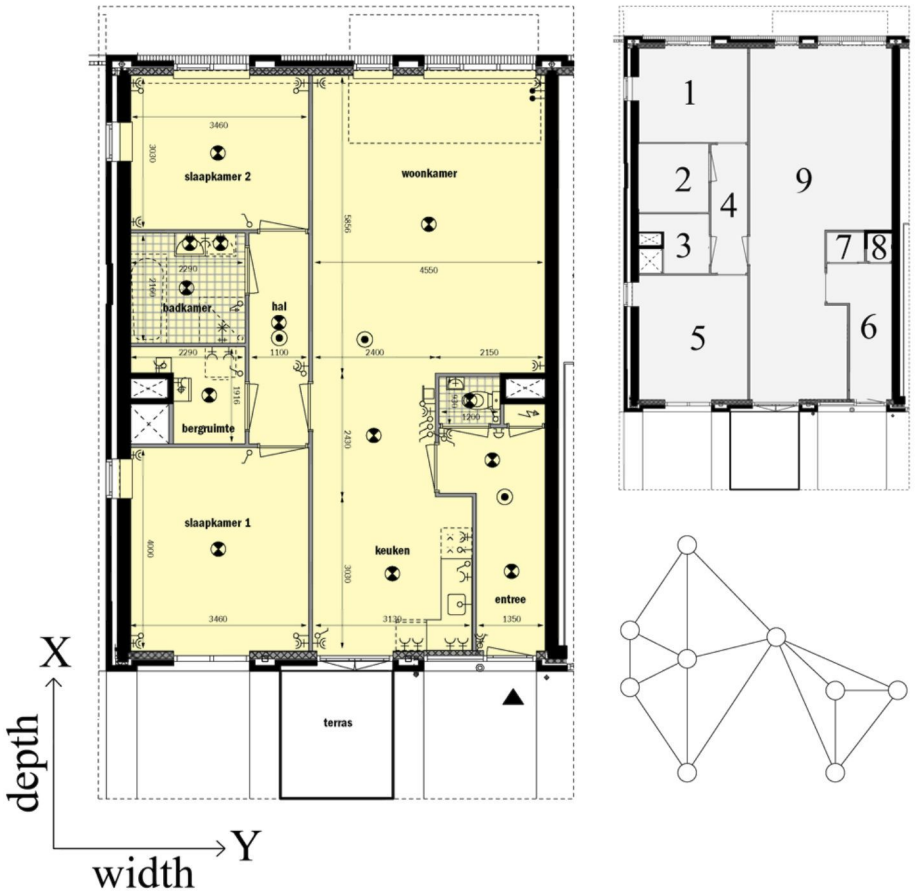

Fig. 1. Original floor plan layout (edited to highlight spaces) and its corresponding space topology graph.

Autodesk's Revit® 2011 works based on families. A family can be roughly described as a set of elements -not necessarily geometries- with an associated set of parameters. Variations of these elements within a family are called "family types", and during this work are equivalent to instantiations of the design solution space. This model is composed by 2 families : the first one defines the geometries and its constraints, and the second is a family of tags used to explore the model by manipulating the modifiable values. Constraints have been previously described as written rules (Table 2) and then translated manually to the model by using the embedded macros tools, as follows :

Table 2. Set of dimensional constraints for each space. Units correspond to [cm].

Space 1	180 < Width < 600; Depth = 305
Space 2	Width = 230; Depth = 215
Space 3	Width = 230; Depth = 190
Space 4	90 < Width < 200; Depth = 405
Space 5	180 < Width < 400; Depth = 400
Space 6	180 < Width < 330; Depth = 447
Space 7	Width = 120; Depth = 93
Space 8	Width = 85; Depth = 93
Space 9	Resulting Space

Constraints were made explicit during the floorplan design process, by adding user-driven text tags in the graphic-user interface, and a real-time 3D visualization of the building (Figure 2). Even though Autodesk's Revit® 2011 has not got a generative design tool for finding every possible solution to a design problem, its capability of differentiating instances using family types allows the user to explore more than one design alternative in the same view, updating at the same time the building information for each solution.

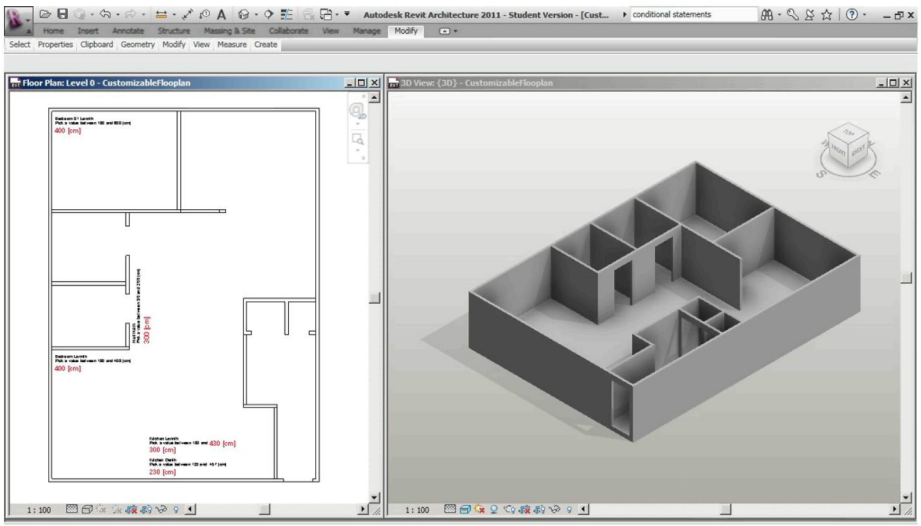

Fig. 2. Graphic-user interface.

4. Constraint model

4.1. Constraint implementation

Autodesk's Revit® 2011 has not got an explicit constraint manager or visual programming interfaces. Constraints are implemented in the model by using conditional statements defined on its macros scripting environment by using the following syntax :

Parameter=IF(Condition, Result-if-true, Result-if-false)

This model considers a parameter and a solution domain interval defined by its minimum and maximum values. For example, every "Length" value is defined by a user-driven parameter "UserLength". Every value lower than the minimum value is automatically constrained to the minimum value "MinLength", and every value higher than the maximum value is constrained to the maximum value "MaxLength", i.e. the correct syntax for setting the minimum and maximum values of the boundaries of the solution domain for this "Length" parameter is :

Length= IF(UserLength<MinLength, MinLength, IF(UserLength>MaxLength, MaxLength, UserLength))

Therefore, the syntaxes for describing the dimensions of the spaces are written as follows (Table 3) :

Table 3. Syntaxes for constraining the dimensions of spaces. Units correspond to [cm].

Space 1	
$Length_X$	= 305
$UserLength_Y$	= $y \in \{R\}$
$MinLength_Y$	= 180
$MaxLength_Y$	= 600
$\Rightarrow Length_Y$	= IF($UserLength_Y$<$MinLength_Y$, $MinLength_Y$, IF($UserLength_Y$> $MaxLength_Y$, $MaxLength_Y$, $UserLength_Y$)) = IF({$y \in \{R\}$}<180, 180, IF({$y \in \{R\}$}>600, 600, {$y \in \{R\}$}))
Space 2	
$Length_X$	= 215
$Length_Y$	= 230

Space 3	
$Length_X$	= 190
$Length_Y$	= 230
Space 4	
$Length_X$	= 405
$UserLength_Y$	= $y \in \{R\}$
$MinLength_Y$	= 90
$MaxLength_Y$	= 200
$\Rightarrow Length_Y$	= IF($UserLength_Y$<$MinLength_Y$, $MinLength_Y$, IF($UserLength_Y$> $MaxLength_Y$, $MaxLength_Y$, $UserLength_Y$)) = IF({$y \in \{R\}$}<90, 90, IF({$y \in \{R\}$}>200, 200, {$y \in \{R\}$}))
Space 5	
$Length_X$	= 400
$UserLength_Y$	= $y \in \{R\}$
$MinLength_Y$	= 180
$MaxLength_Y$	= 400
$\Rightarrow Length_Y$	= IF($UserLength_Y$<$MinLength_Y$, $MinLength_Y$, IF($UserLength_Y$> $MaxLength_Y$, $MaxLength_Y$, $UserLength_Y$)) = IF({$y \in \{R\}$}<180, 180, IF({$y \in \{R\}$}>400, 400, {$y \in \{R\}$}))

Space 6	
$Length_X$	= 447
$UserLength_Y$	= $y \in \{R\}$
$MinLength_Y$	= 180
$MaxLength_Y$	= 330
$\Rightarrow Length_Y$	= IF($UserLength_Y$<$MinLength_Y$, $MinLength_Y$, IF($UserLength_Y$> $MaxLength_Y$, $MaxLength_Y$, $UserLength_Y$)) = IF({$y \in \{R\}$}<180, 180, IF({$y \in \{R\}$}>330, 330, {$y \in \{R\}$}))
Space 7	
$Length_X$	= 93
$Length_Y$	= 120
Space 8	
$Length_X$	= 93
$Length_Y$	= 85

In addition to that, minimum and maximum values can be programmed as the result of sequential operations. Its modification is also calculated in real time but requires bigger processing and graphic capabilities.

This rudimentary definition allows only establishing minimum and maximum values, but despite this the complete data structure can be described by using this syntax. A bigger challenge is the constraint management, as Autodesk's Revit® 2011 does not have a unique interface for its management or visualization, and every constraint must be programmed independently.

4.2. Constraint satisfaction

The main task during the solution finding process is to satisfy every constraint simultaneously. Conversely, in our model constraint satisfaction is directly linked with the conditional visualization of some physical elements, so different alternatives can be explored even though some constraints are not satisfied. For this specific model, boolean statements are used to hide/unhide physical enclosures whilst the user-driven inputs are outside the solution domain for each constraint, i.e. for hiding or unhiding a wall based on its length, the correct syntax for the "WallVisibility" parameter is :

WallVisibility= UserLength>MaxLenght

This feature can be roughly considered as a disjunctive constraint. The addition or elimination of enclosures could be interpreted as disjunctions of the solution-finding process, therefore the solution topology changes.

4.3. User-driven variations

User-editable tags are based on a "Tag" Family that makes explicit the modifiable parameters in the Floorplan view. By clicking on each tag, the user can easily modify the parameters on the Floorplan and the 3D View simultaneously (Figures 3 and 4). By using this feature, different design solutions can be modified independently in the same user view. Each solution is an instance of the solution domain, and consists on the combination of 2 family types : geometries and tags.

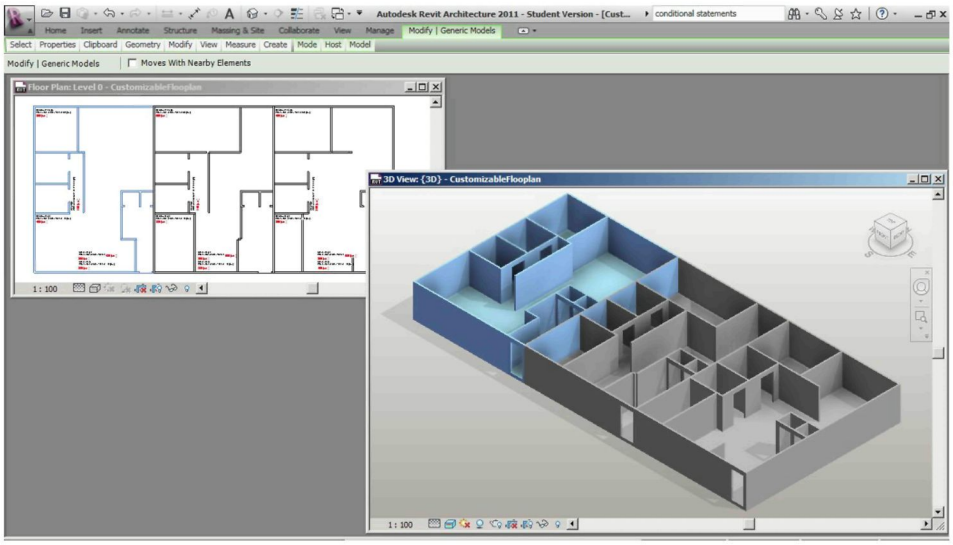

Fig. 3. Three different instances of the problem solution in the same user view.

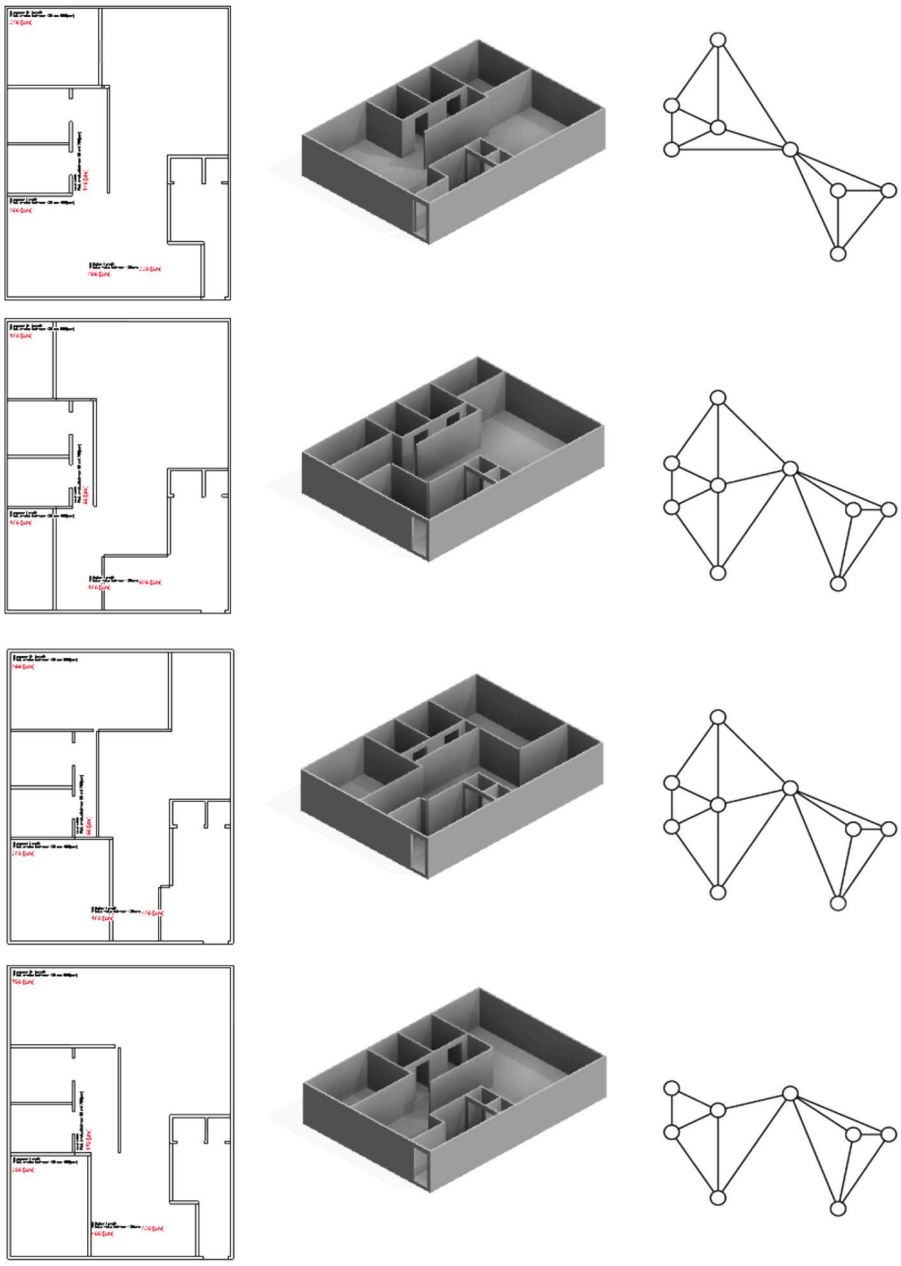

Fig. 4. Four different instances of the spatial layout explorer with the corresponding topology graphs. The nodes that correspond to the sanitary cores have been drawn in a fixed position.

4.4. Limitations

The aim of this model is to demonstrate that a basic constraint-based system can be implemented on commonly used software, with no need of hard text-based programming. However, we have identified some limitations both during the programming and the design exploration processes :

- There is no explicit visualization and management of the constraint system or the data structure of the design problem, for example, as topology graphs or visual programming interfaces. This requires an excellent previous understanding of the design problem and the parameters which are involved on the solution finding process.

- This model has 19 variables -including the visibility constraints-, and 12 constant parameters. Even though a "barrier of complexity" has not been defined yet, the amount of information and required processing is already high, and additional variables may turn the model inefficient or difficult to manipulate.
- A particular feature of this model is that constraints are applied on the dimension and position of physical enclosures, therefore the spatial configuration is not an initial condition but a result of the design exploration.
- The openness of the solution space allows redefining the initial conditions of the design problem by changing the amount of spaces and its topological arrangement. This turns difficult to systematize the solution finding process.

5. Discussion and further work

The macros technology embedded in Autodesk's Revit 2011 used in this context allows the user to participate directly on a project development as a solution finder during early stages of the design process. Despite this, some training is required to implement more consistent solutions based in building restrictions, the formulation of disjunctive/conjunctive constraints, and to systematize the solution finding process. The building information of the model is provided by the software's capabilities, as well as the generation of the technical documentation for different design solutions.

The three most interesting features of this model are : 1) the capability of exploring more than one design solution in the same user view. This feature places this model as an intermediate point between the automated generative design engines that allow the designer to visualize every possible solution, and the one-at-the-time visualization of design outcomes, i.e. Grasshopper; 2) the introduction of an alternative for a friendly graphic-user interface not just for end

users, but also for designers. According to Franke and Piller [28], the "main part of the interaction between the user and the productive system takes place during the configuration process". In this case, the configuration software is the feedback tool itself. Finally, the third feature is 3) the possibility of arranging spaces of irregular shapes by using the visibility commands, increasing the amount of possible variations. This automatically avoids to over-constraint the model and "dead ends" can be hardly found during the solution finding process. These three findings may lead to partially bridge the gap between the explicit knowledge from research and the industry.

Further research work is focused on the development of guidelines for improving modeling processes in architectural design, concerning both virtual and physical models. The presented model is purely virtual and describes some advantages of this realm : a fast instantiation and exploration, and its dialog with other virtual tools for the completion of the design process i.e. the generation of building documentation.

6. Acknowledgements

This work is part of the first author doctoral research at the University of Salford. This research is supported by the "Graduate Teaching Assistantship" program.

References

1. Gropius, W. (1956). The scope of total architecture. Allen and Unwin Ltd.
2. Larson, K. (2000). The home of the future. A+U 361(Oct 2000).
3. Sousa, J.P. & Duarte, J.P. (2005). Digital Desires, Material Realities : Perceiving the technological gap. In J.P. Duarte (Ed.) Proceedings of the eCAADe 2005 Conference : Digital Design : The Quest for New Paradigms. Sept 21-24, Technical University of Lisbon, Portugal.
4. Turk, Z. (2001). The reasons for reality gap in CAAD. In H Penttila (Ed.) Proceedings of the eCAADe 2001 Conference : Architectural Information Management. Aug 29-31, Helsinki University of Technology, Finland.
5. Dave, B. & Koskela, L. (2009). Collaborative knowledge management – A construction case study. Automation in Construction 18(2009) : 894-902.
6. Duarte, J.P. (2005). A discursive grammar for customizing mass housing : the case of Siza's houses of Malagueira. Automation in Construction 14 : 265-275.
7. Gero, J. & Sosa, R. (2008). Complexity measures as a basis for mass customization of novel designs. Environment and Planning B : Planning and Design 35(1) : 3-15.
8. Niemeijer, R.A., de Vries, B. & Beetz, J. (2009). One Size Fits None – A user interface for constraint-based design. In Proceedings of the ACADIA09 Conference : reForm() – Building a Better Tomorrow. Oct 22-25, Chicago, Illinois, United States.

9. Medjdoub, B. & Yannou, B. (2000). Separating topology and geometry in space planning. Computer-Aided Design 32 : 39-61.
10. Medjdoub, B., Richens, P. & Barnard, N. (2002). Generation of variational standard plant room solutions. Automation in Construction 12 : 155-166.
11. Donath, D. & Gonzalez, L.F. (2008). Constraint-based design in participatory housing planning. International Journal of Architectural Computing 6(1) : 97-117.
12. Veliz, A., Gonzalez, L.F. & Barros, L.P. (2008). Design the componentes construtivos usando um metodo de desenho baseado nas restricoes (Design of constructive components using a constraint-based method). Pesquisa em Arquitectura e Construcao 1(3).
13. Niemeijer, R.A., de Vries, B. & Beetz, J. (2009). Check Mate : Automatic constraint checking of IFC models. In A. Dikbas, E. Ergen & H. Giritli (Eds.). CIB W78 : Managing IT in Construction, London, CRC Press, pp. 479-486.
14. Smith, E.A. (2001). The role of tacit and explicit knowledge in the workplace. Journal of Knowledge Management 5(4) : 311-321.
15. Lee, J.Y. & Kim, K. (1996). Geometric reasoning for knowledge-based parametric design using graph representation. Computer-Aided Design 26(10) : 831-841.
16. Niemeijer, R.A., de Bries, B. & Best, J. (2008). Identifying technical obstacles for a constraint-based mass customization system. In H.J.P. Timmermans & B. de Vries (Eds.). Design & Decision Support Systems in Architecture and Urban Planning. Eindhoven.
17. Calderon, C. & Cavazza, M. (2001). Intelligent Virtual Environment for Building Design. In Proceedings of the 5th World Multi-Conference on Systemics, Cybernetics and Informatics. Jul 22-25.
18. Damski, J.C. & Gero, J. (1997). An evolutionary approach to generating constraint-based space layout topologies. In R. Junge (Ed.). Proceedings of the CAAD Futures 1997 Conference. Kluwer, Dordrecht.
19. Eggink, D., Gross, M.D. & Do, E. (2001). Smart Objects : Constraints and Behaviours in a 3D design environment. In H. Penttila (Ed.). Proceedings of the eCAADe 2001 Conference : Architectural Information Management. Aug 29-31, Helsinki University of Technology, Finland.
20. de Vries, B., Jessurun, A.J. & Kelleners, R.H.M.C. (2000). Using 3D geometric constraints in architectural design support systems. In Proceedings of the 8th International Conference in Central Europe in Computer Graphics, Visualization and Interactive Digital Media. Feb 7-10, University of West Bohemia, Czech Republic.
21. Donath, D. & Lobos, D. (2008). Massing study support. In Proceedings of the eCAADe 2008 Conference : Architecture in Computro. Sept 17-20, Antwerpen, Belgium.
22. Aldanondo, M., Hadj-Hamou, K., Moynard, G. & Lamothe, J. (2003). Mass customization and configuration : Requirement analysis and constraint-based modeling propositions. Integrated Computer-Aided Engineering 10 : 177-189.
23. Anantha, R., Kramer, G. & Crawford, R. (1996). Assembly modelling by geometric constraint satisfaction. Computer-Aided Design 28(9) : 707-722.
24. Gross, M. (1996). Elements that follow your rules : Constraint-based CAD layout. In Proceedings of the ACADIA'96 Conference, Tucson, AZ.

25. Farenzena, M. & Fusiello, A. (2009). Stabilizing 3D modeling with geometric constraints propagation. Computer Vision and Image Understanding 113(2009) : 1147-1157.
26. Feng, C.X. & Kusiak, A. (1995). Constraint-based design of parts. Computer-Aided Design 27(5) : 343-352.
27. Bouma, W., Fudos, I. & Hoffmann, C. (1995). A geometric constraint solver. Computer-Aided Design 27(6) : 487-501.
28. Franke, N. & Piller, F. (2002). Configuration toolkits for mass customization – Setting a research agenda. Arbeitsbericht Nr. 33 (Okt. 2002) des Lehrstuhls für Allgemeine und Industrielle Betriebswirtschaftslehre der Technischen Universität München.