IJAC 6-1_proof 4 12/05/08 3:30 pm Page 97

Constraint-Based Design in Participatory Housing Planning

Dirk Donath and Luis Felipe González Böhme

Constraint-Based Design in Participatory Housing Planning

Dirk Donath and Luis Felipe González Böhme

We introduce some novel ideas for a constraint-based design strategy to support participatory housing planning processes in Latin America. The following lines intend to open the discussion on the requirements and effect of the computer implementation of a constraint satisfaction approach to solve elementary design problems in architectural practice. The case study applies to the building massing design problem posed by the Chilean urban regulatory framework that addresses single-family affordable houses. Two different implementation criteria are being tested in an ongoing series of trials providing further considerations. One prototype uses MAXON's CINEMA4D XPRESSO ® visual scripting environment to set up a semi-automated design environment which allows users to edit one feature-based 3D model of massing alternative at a time. The other prototype uses ILOG's OPL STUDIO ® constraint programming environment to achieve fully automated search and 2D visualization of all possible solution alternatives to separate subdomains of the building massing design problem.

1. Introduction

User participation in planning and decision-making processes represents a key contribution to the achievement of the poverty reduction targets set for 2015 by the United Nations [1]. Self-management housing, in its many variants, has proved cost-effectiveness in reducing the shortage and increasing the quality of low-income housing in many developing countries. The call for dwellers collaborative involvement in Chile – Latin America's oldest housing policy [2] – goes back to the early 1950s [3], becoming stronger today by means of the nationwide Housing Solidary Fund [4] support. Chilean participatory model addresses poor households by running exclusively on a competitive project-based funding scheme. Housing subsidy applicants are expected to organize themselves and to develop a concrete project with the collaboration of an authorized NGO. Thus, dweller-planner teams compete against each other for the financial backing of their own developed project. Project goals may range from participatory urban upgrade to the production of new housing estate. Until now, the 85% of all funded projects correspond to the construction of housing units upon new land subdivisions [5]. Most of them belong to some type of progressive-development housing program by which dwellers are exclusively responsible for the progressive enlargement of a small (6 to 30sqm) starter dwelling unit. Especially in Chile, progressive-development housing programs turned into the only affordable housing alternative for the poorest quintile of the population. Indeed, 59.868 dwelling units of this type were finished within the last fifteen years [6] and 215.000 additional units are planned to be finished for 2010 (EL SUR, April 26th, 2006). Until 2001, technical support was allocated at a rate of 4 experts per 400 households [7], today NGOs are collaborating with individual groups of a maximum of 60 households per project [8]. The current scheme reveals itself as the only way for the poorest households to access homeownership in the future, and probably not only in Chile. However, the definition of user participation is quite vague yet, even to many state agents [9]. Unesco [10] encourages the governments to simplify the current criteria, standards, requirements, language and procedures in order to achieve truly participatory and inclusive decision-making processes at all levels. In this highly competitive scenario, efficiency regarding time and cost of planning directly affects dwellers' quality of life, especially when adequate shelter is lacking during the project development time. The early incorporation of regulatory requirements into a design project certainly allows it to stand a better chance in the competition for the financing funds. Additionally, lawful design solutions avoid delays in the permit process. Most elementary problems of architectural design involve expert tasks demanding a considerably large domain-specific knowledge base. Particularly this issue is what hinders true layman participation in the design process. Geometric reasoning and decision-making processes which are carried out at early stages of architectural planning may demand great communication and coordination efforts between planner and user. Empirical evidence shows that software functionalities of the traditionally used architectural CAD-systems are unable to support the systematic and logical exploration of design solution alternatives.

2. The building massing design problem

In the architecture practice, the term "massing" is used to describe the three-dimensional size and shape distribution of buildings or other structures, and their relationships to each other and to open areas and lot lines. The general purpose of the combined effect of all massing regulations is to improve urban habitability standards by guaranteeing basic quality criteria, like e.g. proper solar access, ventilation and sufficient open space concerning inhabited spaces. Massing regulations are part of the zoning regulation which is a key tool for carrying out every urban planning policy.

2.1 Planning and zoning regulation

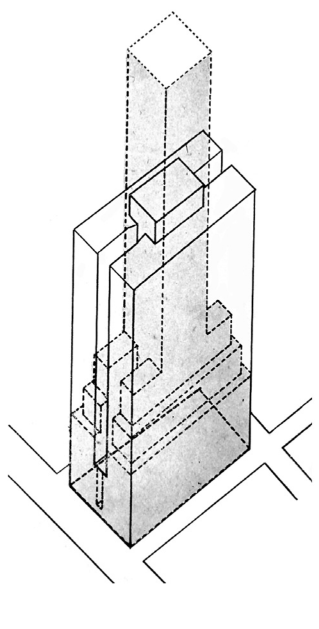
According to the Chilean planning and zoning regulation, the maximum three-dimensional margin, i.e. "maximum envelope", allowable on each zoning lot has to result from applying the pertinent normative constraints over lot lines, sky exposure planes, setbacks, heights, lot coverage, floor area ratio, density and any additional local zoning regulation including the respective special benefit norms in each case [11]. Massing regulations do not work in isolation, they all are related to each other with distinct dependencies, turning the solution-finding process into an expert task. The minimum number of massing regulations and specifications required to control maximum three-dimensional volume of the average type of single-family affordable house currently produced in Chile includes:

1. The building cluster type predetermines an exclusive set of duties and rights applicable to the building on the zoning lot, including three:
   The attached building type which abuts two side lot lines and is one of a row of buildings on adjoining zoning lots.
   The detached building type which is a freestanding building that does not abut any other building on an adjoining zoning lot and where all sides of the building are surrounded by yards or open areas within the zoning lot.
   The semi-detached building type which is a building that abuts or shares one side lot wall with another building on an adjoining zoning lot and where the remaining sides of the building are surrounded by open areas or street lines.
2. The lot coverage ratio is a number which, when multiplied by the lot area of the zoning lot, fixes the maximum amount of ground floor

area allowable in a building on the zoning lot.

3. The floor area ratio is a number which, when multiplied by the lot area of the zoning lot, fixes the maximum amount of floor area allowable in a building on the zoning lot.
4. The building height determines the maximum vertical distance allowable between the natural ground level and the highest point of the building.
5. The story height determines the minimum vertical distance allowable between the finished floor surface and the finished ceiling surface immediately above.
6. The setback determines the minimum horizontal distance between a lot line and the nearest point of the building or any projection thereof, excluding projecting roof structures, eaves, beams, windowboxes or canopies.
7. The zero lot line length ratio is a number which, when multiplied by the length of a side lot line (or a portion thereof) that abuts an adjoining zoning lot, fixes the maximum horizontal distance allowable directly on that side lot line, provided that none of the respective transverse setback lines is trespassed by the building.
8. The zero lot line height determines the maximum vertical distance allowable between the natural ground level or the mean level between adjoining zoning lots and the highest point of the zero lot line building facade.

The structure of the building massing design problem reveals a complex system of the non-redundant combination of all applicable massing regulations.


▶ Figure 1. The Equitable Building, the first modern paradigm of building and zoning regulations.

The level of effectiveness of the solution-finding process mostly depends on the size of the planner's domain-specific knowledge base and her/his spatial reasoning capacity.

2.2 Conventional solving procedures

There are basically two solution-finding methods the architects are used to apply: (a) a top-down method we may call "reductive" and (b) a bottom-up method we may call "constructive". The reductive method consists of sketching a the maximum envelope and then to "prune" it by replacing the original configuration values with some allowed by the applicable massing regulations. A popular and useful heuristics is to choose first extreme values, wherewith the search space becomes considerably smaller. The constructive method instead, begins with directly choosing allowed values for each configuration variable of different parts of the putative building. These parts coincide with those into which the massing regulation itself semantically decomposes the building volume on a zoning lot, like e.g. a zero lot line part of the building or a second floor detached part thereof. In any case, both methods include an evaluation cycle during which some tradeoffs must be made between the assigned values that specify the shape distribution among the building parts. Whatever the method may be, the conventional procedure to produce legal solutions for common building massing design problems consists of the following general steps:

1. Create an instance of solution to the building volume (or part of it) by assigning a value to each configuration variable that specifies that instance in the three-dimensional space unambiguously.
2. Loop until every assigned value are labeled LEGAL:
   (a) Verify whether the assigned value violates any massing regulation.
   i. If the assigned value does not violate any massing regulation, label it as LEGAL.
   ii. Otherwise, assign a new value and return to step (a).
3. Evaluate the legal solution to the building volume according to additional criteria (e.g. cost, aesthetics, etc):
   i. If the legal solution does satisfy most of those criteria, quit.
   ii. Otherwise, return to step 1.

Eventually, step 3 may precede step 2. In any case, the exhaustive verification loop (here, at step 2) may consume a lot of time. In participatory housing planning, the availability of time, human and technical resources to try additional solution alternatives is rare.

2.3 Constraint-based design strategy

Whereas constraints are unfairly identified with restriction and limitation in design [12], there is informal empirical evidence in architectural practice and scientific evidence of significant CAAD research contributions, like e.g. [13; 14; 15; 16; 17; 18 and 19] that constraint-based design may offer a solid formal framework for systematic and logical design space exploration in architecture. Up to now, all scientific evidence point to the use of constraint-satisfaction techniques only as a special purpose tool exclusively focused on the support of narrow design tasks accomplished within the architectural design process. The philosophy underlying the constraint-based design approach is specifically based on the paradigm of constraint satisfaction problems (CSP):

> "A constraint satisfaction problem is defined by a set of variables, $X_1, X_2 \dots X_n$, and a set of constraints, $C_1, C_2 \dots C_m$. Each variable X_i has a nonempty domain D_i of possible values. Each constraint C_i involves some subset of the variables and specifies the allowable combinations of values for that subset. A state of the problem is defined by an assignment of values to some or all of the variables, $\{X_i = v_i, X_j = v_j, \dots\}$. An assignment that does not violate any constraints is called a consistent or legal assignment. A complete assignment is one in which every variable is mentioned, and a solution to a CSP is a complete assignment that satisfies all the constraints." [20]

Thus, we may actually formalize any elementary design problem using the following construct:

$\langle X, D, C \rangle$

where X = a finite sequence of variables $\{x_1, x_2 \dots x_n\}$;
D = a sequence of all possible values that can be assigned to each variable in X;
C = a finite (possibly empty) set of constraints over an arbitrary subsequence of variables in X.

The task in a CSP is to satisfy all constraints simultaneously by assigning a value to each variable of the problem. Architectural planning is basically a two-step problem solving process consisting of programming and design. The task of architectural programming is to specify a set of constraints or "conditions" describing the design problem in as accurate a way as possible. The aim of architectural programming is to provide a specifically enough statement of the given design problem [21]. As mentioned above, the architectural design task is to satisfy all constraints describing the problem structure. In this sense, architectural design may also be considered as a two-step process consisting of propagating the available constraints throughout the implicit network of dependencies between them, and to search for the or a solution by introducing new constraints to the problem. Dependencies between constraints are usual, and the reason for that is because many constraints involve more than one variable of the problem and many variables participate in more than one constraint [22]. As constraints are propagated, additional constraints may be inferred from those that were originally given as part of the problem statement. That leads to a reduction of the domain of possible values for many variables. Constraint propagation terminates for one of two reasons: (a) a contradiction may be detected, e.g. because the problem has been over-constrained, or (b) it becomes impossible to make further changes on the basis of the current knowledge, whether any solution has been found or not [22]. If no solution has been found thereafter, search may begin. Search demands to make a guess about a value for a particular variable which may lead either to a contradiction (inconsistency) or to the generation of additional constraints. Notice that if the design does not provide any convincing solution or at least a legal one, programming begins again and so on. In such terms, architectural planning may consist of the following general steps:

1. Architectural programming.
   - (a) Identify the minimum number of variables required to describe the design problem and to define a complete solution to that problem.
   - (b) Specify a number of constraints on subsequences of these variables.
2. Architectural design.
   - (a) Propagate the available constraints until the domain of every known variable has been reduced to its minimum range:
   - i. If the combined effect of all constraints defines a solution, quit and proceed to evaluate that solution.
   - ii. If the combined effect of all constraints defines an inconsistency, report failure and return to step 1.
   - iii. Otherwise, proceed to step 2(b).
   - (b) Search until a solution is found or all possible solutions have been discarded:
   - i. Select a variable whose value is not yet determined, strengthen as much as possible the set of constraints that apply to that variable and return to step 2(a).

Modeling a problem by means of constraints is natural to the architect, when it comes to describing as much the complete problem as a partial solution of it. The more accurately the problem structure is described, the easier it will be to solve it. In these terms, the act of designing may be simplified to the straightforward assignment of values to a well-defined sequence of problem variables. The sole task of assigning values is not further considered an expert task and therefore it can be shared and discussed by laymen and experts concurrently. That is how constraint-based design may actually support truly participatory planning at all levels.

3. Computer implementation strategy

We are trying different implementation strategies, considering all aspects informed above. The strategy of mechanical problem-solving requires defining a problem structure accurately enough to reduce the problem to a selection problem [23]. The most important advantage of the CSP model is that its standard representation already reveals the structure of the problem itself [20]. Since the search space containing all the conceivable design solution alternatives is too big to be explicitly represented, the solution space can be implicitly represented by means of a constraint system. Kramer [24] defines a constraint system as the collection of geometric entities and the constraints that describe how these entities interact with each other. The task in a geometric CSP is to find the positions, orientations, and dimensions of these geometric entities, so as to satisfy all constraints simultaneously. We adopted Kramer's point of view but renaming the geometric entities as "units". As known to the authors, Tobin's [15] research is the closest related work in the field of constraint-based design strategies applied to tackle zoning regulation problems in architectural design.

3.1 Mathematical model

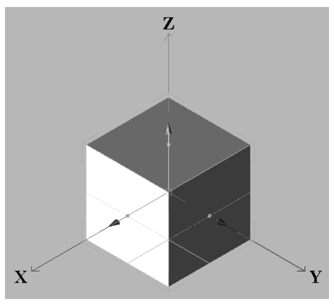
In this section, we introduce a piece of the mathematical model we developed, in order to provide an overview of the methodology we adopted to achieve, in a relative comprehensible way, a formal representation of the variables and constraints involved in many elementary design problems, especially in the building massing design problem.

Elementary variables

Consider the sequence of six configuration variables $P_{x_i}, P_{y_i}, P_{z_i}, S_{x_i}, S_{y_i}, S_{z_i}$ as the minimum number of real-valued parameters required to specify each unit i, $i \in \{1, \ldots, n\}$ in the three-dimensional space unambiguously. The subsequence represents the three translational DOFs (Degrees of Freedom) and the subsequence $S_{x_i}, S_{y_i}, S_{z_i}$ the three dimensional DOFs for each unit i involved in the design problem. The unit U_i is only defined by the sequence of six configuration variables $P_{x_i}, P_{y_i}, P_{z_i}, S_{x_i}, S_{y_i}, S_{z_i}$.

$$U_i = \{P_{x_i}, P_{y_i}, P_{z_i}, S_{x_i}, S_{y_i}, S_{z_i}\} \qquad (1)$$

where the triple (x, y, z) represents a subset of the real coordinate space $\Re^3$. The Euclidean distance is considered. We establish the origin of the coordinate system $x_i\, y_i\, z_i$ in the centroid of each unit Ui, $\forall i \in \{1, \ldots, n\}$. A detailed explanation of this option and other relevant alternatives are cited by Michalek [25].



◀ Figure 2. Graphic representation of a Unit.

Due to the different technical requirements of each prototypical implementation environment we are trying, we establish a flexible definition which may be easily adapted:

Let be $$S^*_{w_i} = \frac{S^*_{w_i}}{2}, w \in \{x, y, z\} \qquad (2)$$

Inferred variables

We may formalize the position of each side of a unit i by means of the following Eqns (3 – 7).

The left side of unit *i* $$P_{Left_i} = P_{x_i} - S^*_{x_i} \qquad (3)$$

The rear side of unit *i* $$P_{Back_i} = P_{y_i} - S^*_{y_i} \qquad (4)$$

The right side of unit *i* $$P_{Right_i} = P_{x_i} - S^*_{x_i} \qquad (5)$$

The bottom side of unit *i* $$P_{Bottom_i} = P_{z_i} - S^*_{z_i} \qquad (6)$$

The top side of unit *i* $$P_{Top_i} = P_{z_i} - S^*_{z_i} \qquad (7)$$

Unit groups

In order to make the representation of architectural spaces more comprehensible, we define the following Eqns (8 – 12).

The floor area of unit i $$A_{Floor_i} = S_{x_i} \times S_{y_i} \qquad (8)$$

The sequence of ground floor units $$U_{GF} = \{U_1, \ldots, U_{a-1}\} \qquad (9)$$

The sequence of upper floor units $$U_{GF} = \{U_a, \ldots, U_n\} \qquad (10)$$

The total ground floor area
$$A_{U_{GF}} = \sum_{i=1}^{a-1} A_{floor_i} \tag{11}$$

The total upper floor area
$$A_{U_{UF}} = \sum_{i=a}^{n} A_{floor_i} \tag{12}$$

Massing constraints

The following Eqns (13 – 25) express different design constraints concerning massing regulations.

The lot coverage constraint
$$A_{U_{GF}} \leq A_{Lot_i} \times \lambda \tag{13}$$

The floor area constraint
$$A_{U_{GF}} + A_{U_{UF}} \leq A_{Lot_i} \times \kappa \tag{14}$$

The building height constraint
$$P_{Top_{max}} \leq h_{Building} \tag{15}$$

where
$$P_{Top_{max}} = \max_{i \in \{1,\ldots,n\}} h_{Building} \tag{16}$$

The story height constraint
$$h_{Story} \leq S_{x_i} \leq h_{Building} \tag{17}$$

In the following Eqns (18 – 21) , units with the subindex *j* represent the setbacks, according to three height intervals specified by the zoning regulation (h ≤ 3.5m; 3.5m < h ≤ 7.0m; 7.0m < h), while the subindex *i* is assigned to the units that represent architectural spaces i.e. building spatial components.

A front setback constraint
$$(P_{Back_i} \leq P_{Front_j}) \wedge (S_{x_i} = S_{x_j}) \tag{18}$$

A left side setback constraint
$$(P_{Right_i} \leq P_{Left_j}) \wedge (S_{y_i} = S_{y_j}) \tag{19}$$

A rear setback constraint
$$(P_{Back_i} \leq P_{Front_j}) \wedge (S_{x_i} = S_{x_j}) \tag{20}$$

A right side setback constraint
$$(P_{Right_i} \leq P_{Left_j}) \wedge (S_{y_i} = S_{y_j}) \tag{21}$$

In the following Eqns (21 – 24) , units with the subindex *k* represent the zoning lot, while the subindex *i* is assigned to the units that represent a zero lot line building (or building spatial component).

Zero lot line length constraint over the left side of the zoning lot
$$(P_{Left_i} \leq P_{Left_k}) \wedge (S_{y_i} \leq S_{y_k} \times \mu) \leq (\sum S_{y_i} \leq S_{y_k} \times \mu) \tag{22}$$

Zero lot line length constraint over the right side of the zoning lot
$$(P_{Right_i} \leq P_{Right_k}) \wedge (S_{y_i} \leq S_{y_k} \times \mu) \leq (\sum S_{y_i} \leq S_{y_k} \times \mu) \tag{23}$$

Zero lot line length constraint over the rear side of the zoning lot

$$(P_{Back_i} \leq P_{Back_k}) \wedge (S_{y_i} \leq S_{y_k} \times \mu) \leq (\sum S_{y_i} \leq S_{y_k} \times \mu) \qquad (24)$$

where µ is a real value ranging over [0..1] which, when multiplied by the length of a side lot line (or a portion thereof) that abuts an adjoining zoning lot, fixes the maximum horizontal distance allowable directly on that side lot line, provided that none of the respective transverse setback lines is trespassed by the building.

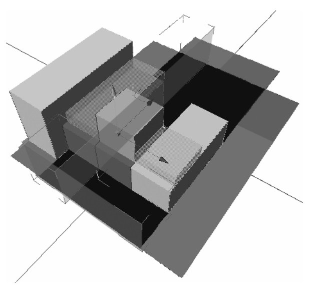
In the following Eqn (25) , units with the subindex *m* represent a possibly existent zero-lot line building unit on an adjoining zoning lot , while the subindex *i* is assigned to the units that represent a zero lot line building (or building spatial component).

The zero lot line height constraint

$$(S_{z_i} \leq h_{Zero}) \vee ((S_{z_i} \leq S_{z_m}) \wedge (h_{Zero} \leq S_{z_m}) \qquad (25)$$

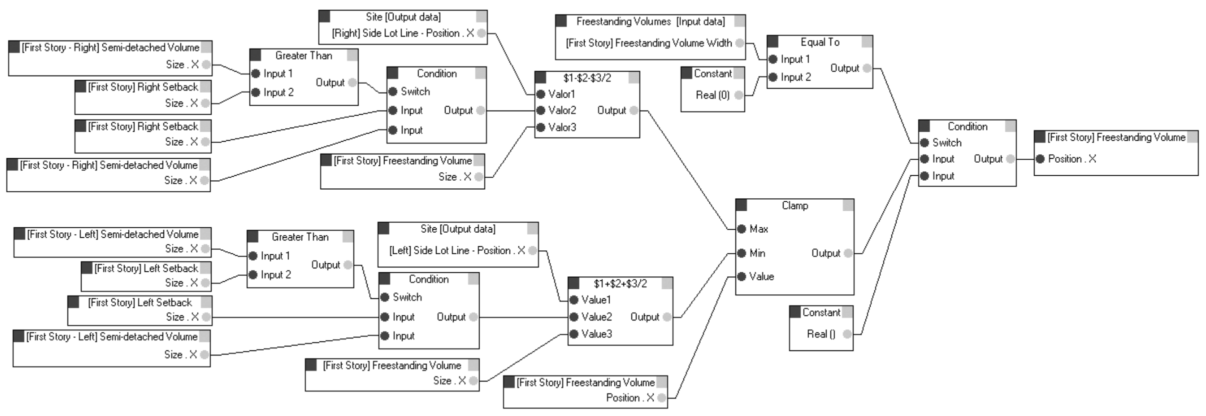
3.2 Prototypes

The implementation of prototype No. 1 on MAXON's CINEMA4D R10 ® supports the constructive method by allowing to edit one parametric feature-based 3D model of a building massing alternative solution at a time. The implementation model poses a minimum number of 25 different units (spatial components) required to describe the massing design problem. We differentiate three classes of units according to what they represent: (a) zoning lots, (b) buildings and (c) setbacks. The graphic representation of the building itself is made up of two types of units: (b1) detached units and (b2) semi-detached units.

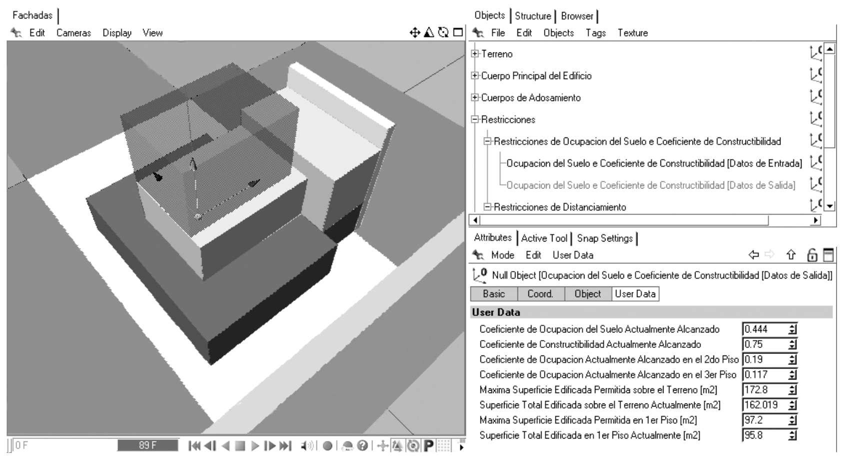

◀ Figure 3. Minimum number of units required.

Modeling the building massing design problem by using Maxon's XPresso® visual scripting language provides an implicit representation of the constrained search space itself. The resulting digraphs representing the constraint system [17] display the high complexity of the problem structure.

The prototype does not allow creating new alternative building massing solutions from scratch, but it certainly supports a systematic exploration of additional alternatives in real time with consistent data feedback. The shape diversity that may be achieved by using this GUI-based application is limited, but still enough to provide efficient support in terms of time and planning cost at early phases of participatory planning sessions.


[First Story - Right] Semi-detached Volume Size . X
[First Story] Right Setback Size . X
[First Story] Right Setback Size . X
[First Story - Right] Semi-detached Volume Size . X
Greater Than
Input 1
Input 2
Output
Site [Output data]
[Right] Side Lot Line - Position . X
Condition
Switch
Input
Input
Output
[First Story] Freestanding Volume Size . X
$1-$2-$3/2
Valor1
Valor2
Valor3
Output
Freestanding Volumes [Input data]
[First Story] Freestanding Volume Width
Constant
Real (0)
Equal To
Input 1
Input 2
Output
Condition
Switch
Input
Input
Output
[First Story] Freestanding Volume Position . X
Clamp
Max
Min
Value
Output
Constant
Real ()
[First Story - Left] Semi-detached Volume Size . X
[First Story] Left Setback Size . X
[First Story] Left Setback Size . X
[First Story - Left] Semi-detached Volume Size . X
Greater Than
Input 1
Input 2
Output
Site [Output data]
[Left] Side Lot Line - Position . X
Condition
Switch
Input
Input
Output
[First Story] Freestanding Volume Size . X
$1+$2+$3/2
Value1
Value2
Value3
Output
[First Story] Freestanding Volume Position . X

▲ Figure 4. Position in X of a detached unit on a zoning lot considering setbacks.


Fachadas
Edit Cameras Display View
Objects Structure Browser
File Edit Objects Tags Texture
Terreno
Cuerpo Principal del Edificio
Cuerpos de Adosamiento
Restricciones
Restricciones de Ocupacion del Suelo e Coeficiente de Constructibilidad
Ocupacion del Suelo e Coeficiente de Constructibilidad [Datos de Entrada]
Ocupacion del Suelo e Coeficiente de Constructibilidad [Datos de Salida]
Restricciones de Distanciamiento
Attributes Active Tool Snap Settings
Mode Edit User Data
Null Object [Ocupacion del Suelo e Coeficiente de Constructibilidad [Datos de Salida]]
Basic Coord. Object User Data
0 F
89 F

User Data

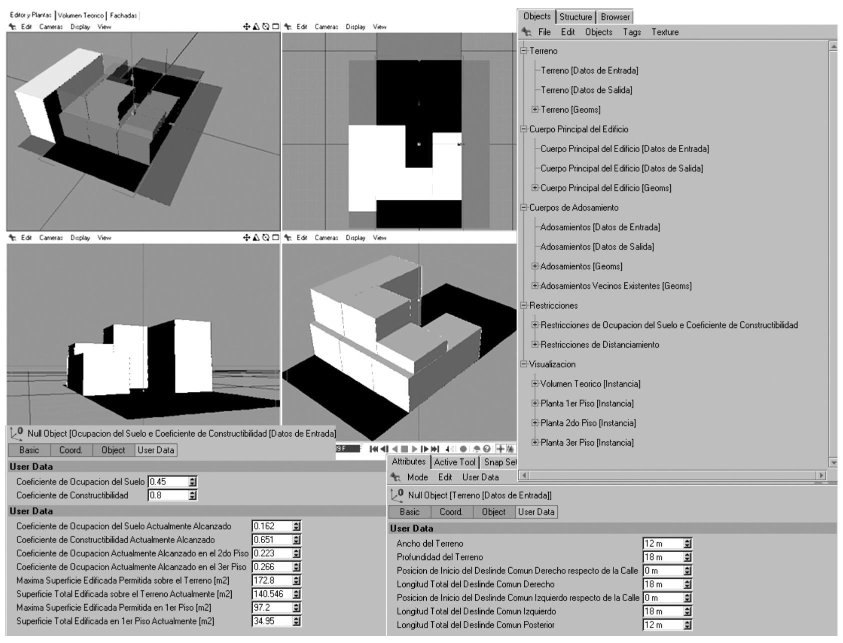
Coeficiente de Ocupacion del Suelo Actualmente Alcanzado	0.444
Coeficiente de Constructibilidad Actualmente Alcanzado	0.75
Coeficiente de Ocupacion Actualmente Alcanzado en el 2do Piso	0.19
Coeficiente de Ocupacion Actualmente Alcanzado en el 3er Piso	0.117
Maxima Superficie Edificada Permitida sobre el Terreno [m2]	172.8
Superficie Total Edificada sobre el Terreno Actualmente [m2]	162.019
Maxima Superficie Edificada Permitida en 1er Piso [m2]	97.2
Superficie Total Edificada en 1er Piso Actualmente [m2]	95.8

► Figure 5. GUI of the prototype running on CINEMA4D ©.

The high-level interaction functionalities provided by MAXON's XPRESSO ® visual scripting environment enabled us to apply a software prototyping methodology which delivered large amounts of high qualitaty data to our research.

The implementation of prototype No. 2 on ILOG's optimization modeling environment OPL STUDIO 3.7.1 ® may only provide two-dimensional graphic representations, but its built-in solver is able to find and display the complete sequence of solution alternatives for a given building massing design problem fully automated. However, up to now, we are only able to solve separate subdomains of the building massing design problem, due mainly to the lack of three-dimensional visualization.

As the opposite of the imperative programming paradigm, where exact step-by-step instructions about what to do have to be given, the constraint programming (CP) paradigm rather focuses on an accurate description of the properties the putative solution shall have and lets the computer work out how to find it by using general purpose techniques. These techniques are applied then by a computing engine called the constraint solver which usually searches for one or more solutions guided by some heuristics we


Editor y Planta | Volumen Teorico | Fachadas
Edit Cameras Display View
Objects | Structure | Browser
File Edit Objects Tags Texture
Attributes | Active Tool | Snap Set
Mode Edit User Data
Basic | Coord. | Object | User Data

Terreno
- Terreno [Datos de Entrada]
- Terreno [Datos de Salida]
- Terreno [Geoms]

Cuerpo Principal del Edificio
- Cuerpo Principal del Edificio [Datos de Entrada]
- Cuerpo Principal del Edificio [Datos de Salida]
- Cuerpo Principal del Edificio [Geoms]

Cuerpos de Adosamiento
- Adosamientos [Datos de Entrada]
- Adosamientos [Datos de Salida]
- Adosamientos [Geoms]
- Adosamientos Vecinos Existentes [Geoms]

Restricciones
- Restricciones de Ocupacion del Suelo e Coeficiente de Constructibilidad
- Restricciones de Distanciamiento

Visualizacion
- Volumen Teorico [Instancia]
- Planta 1er Piso [Instancia]
- Planta 2do Piso [Instancia]
- Planta 3er Piso [Instancia]

Null Object [Ocupacion del Suelo e Coeficiente de Constructibilidad [Datos de Entrada]

User Data

Coeficiente de Ocupacion del Suelo	0.45
Coeficiente de Constructibilidad	0.8

User Data

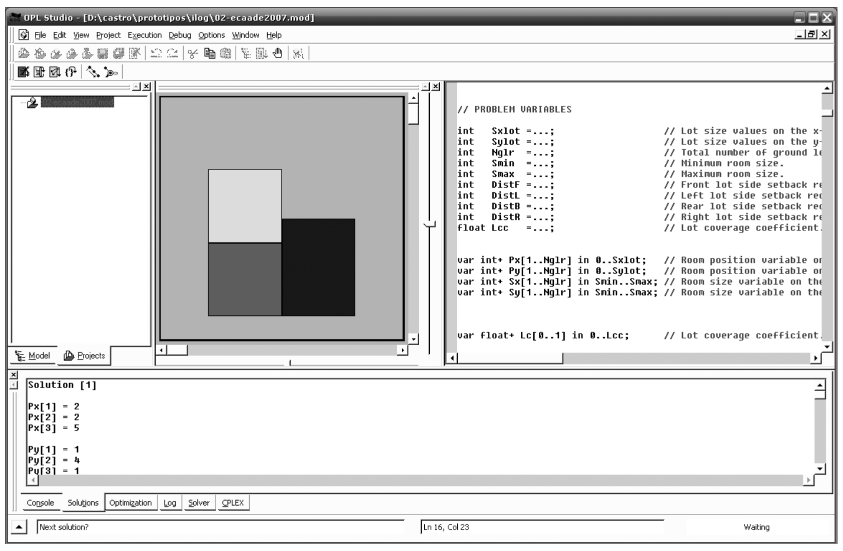
Coeficiente de Ocupacion del Suelo Actualmente Alcanzado	0.162
Coeficiente de Constructibilidad Actualmente Alcanzado	0.651
Coeficiente de Ocupacion Actualmente Alcanzado en el 2do Piso	0.223
Coeficiente de Ocupacion Actualmente Alcanzado en el 3er Piso	0.266
Maxima Superficie Edificada Permitida sobre el Terreno [m2]	172.8
Superficie Total Edificada sobre el Terreno Actualmente [m2]	140.546
Maxima Superficie Edificada Permitida en 1er Piso [m2]	97.2
Superficie Total Edificada en 1er Piso Actualmente [m2]	34.95

Null Object [Terreno [Datos de Entrada]]

User Data

Ancho del Terreno	12 m
Profundidad del Terreno	18 m
Posicion de Inicio del Deslinde Comun Derecho respecto de la Calle	0 m
Longitud Total del Deslinde Comun Derecho	18 m
Posicion de Inicio del Deslinde Comun Izquierdo respecto de la Calle	0 m
Longitud Total del Deslinde Comun Izquierdo	18 m
Longitud Total del Deslinde Comun Posterior	12 m

◀ Figure 6. Feature-based prototype running on CINEMA4D ©.


OPL Studio - [D:\castro\prototipos\ilog\02-ecaade2007.mod]
File Edit View Project Execution Debug Options Window Help
02-ecaade2007.mod
Model Projects
Console Solutions Optimization Log Solver CPLEX
Next solution?
Ln 16, Col 23
Waiting

```
// PROBLEM VARIABLES

int   Sxlot =...;                 // Lot size values on the x-
int   Sylot =...;                 // Lot size values on the y-
int   Nglr  =...;                 // Total number of ground le
int   Smin  =...;                 // Minimum room size.
int   Smax  =...;                 // Maximum room size.
int   DistF =...;                 // Front lot side setback re
int   DistL =...;                 // Left lot side setback rec
int   DistB =...;                 // Rear lot side setback rec
int   DistR =...;                 // Right lot side setback re
float Lcc   =...;                 // Lot coverage coefficient.


var int+ Px[1..Nglr] in 0..Sxlot;   // Room position variable on
var int+ Py[1..Nglr] in 0..Sylot;   // Room position variable on
var int+ Sx[1..Nglr] in Smin..Smax; // Room size variable on the
var int+ Sy[1..Nglr] in Smin..Smax; // Room size variable on the



var float+ Lc[0..1] in 0..Lcc;      // Lot coverage coefficient.
```

```
Solution [1]

Px[1] = 2
Px[2] = 2
Px[3] = 5

Py[1] = 1
Py[2] = 4
Py[3] = 1
```

◀ Figure 7. GUI of the prototype running on OPL Studio.

may specify. Maybe the most important advantage is that the order in which constraints are declared does not matter. A little sample of how constraints may be declared using ILOG's OPL STUDIO 3.7.1 ® follows:

A setback constraint group

- forall(i in 1..Nglr) {
- Px[i] >= DistL;
- Py[i] >= DistF;
- Px[i]+Sx[i] <= Sxlot-DistR;

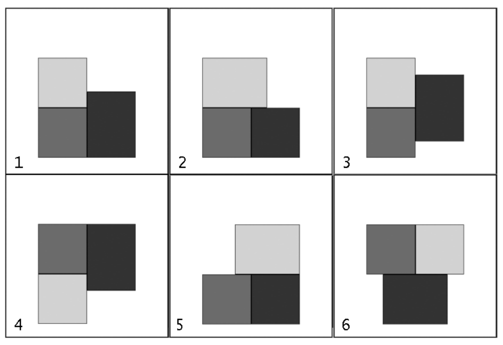
- Py[i]+Sy[i] <= Sylot-DistB;
- };

where the variable Nglr controls the total number of rooms on the ground level; the variables DistL, DistF, DistR and DistB represent the left, front, right and rear setback distances respectively, Sxlot represents the lot width and Sylot the lot depth.

A lot coverage constraint

- sum (i in 1..Nglr) Sx[i]*Sy[i] = (Sxlot*Sylot)*Lcc;

where the real variable Lcc ranging over [0..1] represents the lot coverage ratio.

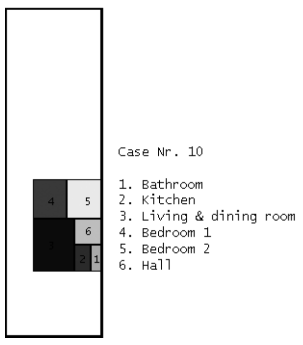


▶ Figure 8. Subsequence of 6 solution alternatives (from a total of 72) found with the OPL Studio prototype.

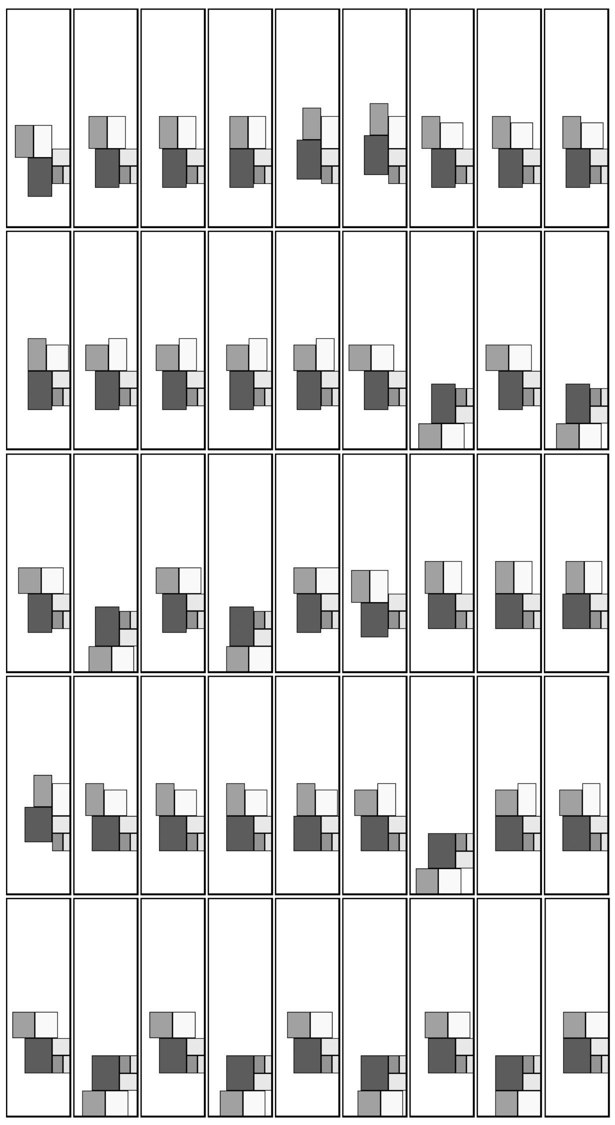
For instance, let us suppose we want to find all possible ways of covering 30% of a 10x10m square zoning lot, having three ground level rooms of variable size, ranging over 3m to 10m. Furthermore, there are setback requirements of 1m to the front, 2m to the left, 3m to the rear side, and 2m to the right side of the zoning lot. We modeled 37 constraints defined on 14 variables and the ILOG's solver found 72 solutions for that specific problem in less than one second by using depth first search.

A different experiment focused on the programming of topological constraints. It was conducted in order to explore what space allocation alternatives a particular household really could have had by the time they started the development of their own house upon a given zoning lot imposed by the "site-and-services" program, back in 1974 in the city of Concepción, Chile. Starting with the current space allocation (last surveyed in 2004), progressively achieved through the household's own effort over thirty years, the adjacency relationships between all dwelling spaces were modeled as a disjunctive group of topological constraints. In this sense, both width and depth values for each dwelling space were considered interchangeable.

IJAC 6-1_proof 4 12/05/08 3:30 pm Page 112



◀ Figure 9. Last development state of the surveyed house (2004).



◀ Figure 10. All possible 45 space allocation alternatives, found by applying 51 constraints over 24 design variables.

Thus, adjacency relationships were considered symmetric and not univocally oriented, i.e. if space A is adjacent to space B, then space A may be located either to the left, to the right, to the front or to the back of space B and vice versa. An additional geometric constraint specifying a minimum contact length between adjacent spaces was introduced so as to allow access to every space represented on the floor plan. In order to increase heterogeneity among the solution alternatives and also to reduce computing time, the domain of each variable was limited to the set of integer values. Additionally, modular values were used. The only fixed sequence of values corresponded to the lot geometry and the position of the 6 sqm starter dwelling unit consisting of a bathroom and cooking area. The solution space was not as large as we thought in the first place and the sequence of space allocation alternatives did not show much diversity among them.

A total amount of 51 constraints over 24 design variables were necessary to model this elementary design problem. The experiment proved great intuition from the household members to preview the real enlargement possibilities resulting from the combination of a set of imposed initial conditions and their own design constraints.

4. Comparative analysis

The two software prototypes described in this article are compared within the frame of constraint-based design theory. The two prototypes were mainly conceived as tools for test and validation of possible computational methods in constraint-based design theory. The prototypes are compared from two different points of view. The first point of view verifies whether each of the prototypes may or may not accomplish the minimum number of design tasks required to specify a building envelope according to an elementary set of zoning regulations.

▼ **Table 1. Prototypes comparison under the point of view of the accomplishment of specific design tasks.**

Item No.	Design tasks	Prototype No. 1 modeled with CINEMA4D ©	Prototype No. 2 modeled with OPL STUDIO ©
A1	Specify zoning lot geometry	Yes	Yes
A2	Specify adjoining zoning lots geometry	Yes	Not yet
A3	Specify attached building shapes	Yes	Yes
A4	Specify detached building shapes	Yes	Yes
A5	Specify semi-detached building shapes	Yes	Yes
A6	Optimize lot coverage	Yes	Yes
A7	Optimize floor area ratio	Yes	No
A8	Specify building height	Yes	No
A9	Specify story height	Yes	No
A10	Specify setbacks	Yes	Yes
A11	Optimize zero lot line building length	Ye	Not yet
A12	Optimize zero lot line building height	Yes	Not yet
A13	Specify neighboring zero lot line buildings	Yes	Not yet

Notice that, by the time this comparative analysis was conducted, the development of prototype No. 1 (implemented on CINEMA4D R10 ©) was already finished, showing all its capacities, whereas the development of prototype No. 2 (implemented on ILOG OPL STUDIO 3.7.1) was in progress, unable to reveal all its capacities yet. Only three aspects, described in items No. A7 to A9, namely the possibility of specifying heights and floor area ratio are already discarded due to the lack of three-dimensional visualization of the development platform on which prototype No. 2 was implemented. The incapacity of prototype No. 2 to accomplish the design tasks announced in the items No. A11 to A13, revealed the need to reformulate the corresponding equations described in the mathematical model. Whereas the implementation platform for prototype No. 1 – a sophisticated geometry modeling and animation software – provides a comprehensive GUI offering high level of interactivity, the implementation platform for prototype No. 2 is a less user-friendly programming environment for modeling and solving optimization problems, which clearly demands some knowledge of computer programming in order to use and manipulate prototype No.2.

The second point of view compares aspects related to both the development process and operational aspects between the two different the prototypes. Whereas the development process of prototype No. 1 demands the exhaustive construction of the entire dependency network between the design variables, the constraint programming model supported by the Optimization Programming Language (OPL) keeps it implicit in the formulation of the design constraints. The symbols "+" and "–" are used to express how each prototype responds with "more" or "less" capacity under different perspectives.

▼ Table 2. Prototypes comparison under the point of view of operational aspects.

Item No.	Perspective	Prototype No. 1 modeled with CINEMA4D ©	Prototype No. 2 modeled with OPL STUDIO ©
01	Automation	–	+
02	Flexibility	–	+
03	Feedback	–	+
04	Visualization	+	–
05	Ease of use	+	–

The development process for prototype No. 1 was extraordinary slower compared to the constraint modeling process carried out for prototype No. 2. The higher level of automation is found in prototype No. 2, mainly provided by the ILOG's solver which conducts automatic search through the solution space. Prototype No. 2 also shows more flexibility because contrary to prototype No. 1, it does not need a pre-existing geometric model to edit, allowing instead the creation of additional units (geometric entities), dimensional and topological variety. Additionally, new constraints may be added or deleted at any time without following any hierarchical order into the programming corpus. The qualitatively best system feedback is found in prototype No. 2 which is able to display all possible solutions to a specific design problem by supporting a comprehendible and traceable design space exploration. In contrast, prototype No. 1 continuously displays the same solution in real time with its corresponding modifications. Data visualization is qualitatively superior in prototype No. 1, mainly because of the comprehensive GUI of CINEMA4D © and the aforementioned lack of three-dimensional display capacity of prototype No. 2. The visual scripting language with which the prototype No. 1 was modeled, allows us to visualize and comprehend the complexity of design problem structures. The emergent paradigm of visual programming languages (VPL) will encourage more architects to develop their own design tools tailored to their specific needs.

5. Conclusion

One of the greatest advantages of the Optimization Programming Language [26] is the fact that it is a declarative language. The declarative nature of constraints has many benefits: the order in which constraints are imposed has no importance and additional constraints, that capture new properties of the solutions, can be added without worrying about the interaction with existing constraints and the search procedure. The contrary case is the prototype implemented on MAXON's CINEMA4D ©, where the constraint system must be modeled as a directed graph.

There is more than one representation of a problem as a constraint satisfaction problem. Constraint programming provides a fully automated procedure to solve a CSP, but it is not the only one. More important than the particular tool to solve a problem seems to be its formal representation, i.e. the result of understanding the problem in depth and be able to explain it to others or to translate it into computer language. The level of effectiveness in using one or the other prototype exposed here strongly depends on the problem structure itself. All told, prototype No. 2, developed with ILOG's OPL STUDIO ® constraint programming environment proves qualitatively superior advantages compared to prototype No1., implemented in CINEMA4D. Computational methods tested on prototype No. 2, proved relevant in applying constraint-based design theory and practice to participatory design practices. The balance between full automation and interactivity is what gives to the designer full control in solving complex problems [27].

Further research work is focused on the elementary design problem concerning dwelling space allocation. We started a series of trials on ILOG's OPL STUDIO ® constraint programming environment based on the work of Loemker [28], particularly on the modeling approach for implementing topological relationships between units. Our next steps address the domain of qualitative spatial reasoning and especially the RCC8 model (region connection calculus).

References

1. United Nations HABITAT, ed., *The Global Campaign on Urban Governance*, United Nations Human Settlements Programme, Nairobi, 2002.
2. Hidalgo, R., La Vivienda Social en Chile: La Acción del Estado en Un Siglo de Planes y Programas, *Scripta Nova*, 1999, vol. 45(1), http://www.ub.es/geocrit/sn-45-1.htm [30-1-2007].
3. Green, M., El Programa de Vivienda Progresiva en Chile 1990-2002, *Inter-American Development Bank Publication*, Social Development Division, 2004.
4. http://www.fsv.cl [16-4-2006].
5. Nieto, M., Intervention by the IV Jornada de Vivienda Social: Síntesis Panel No. 3, *Impacto de la Política Habitacional*, Santiago de Chile, 2005.
6. Ministry of Housing and Urbanism of Chile, *Viviendas Terminadas y Subsidios Pagados: 1990-2005*, Ministerio de Vivienda y Urbanismo de Chile, http//www.minvu.cl [3-12- 2005].
7. Ministry of Housing and Urbanism of Chile, *Programa de Asistencia Técnica Sectorial*, Ministerio de Vivienda y Urbanismo de Chile, Resolución No. 533 de 1997, JTF/AHM/CMR, 31.07.2004.
8. Saborido, M., *Experiencias Emblemáticas para la Superación de la Pobreza y Precariedad Urbana en América Latina y el Caribe*, Economic Commission for Latin America and the Caribbean, http://www.eclac.org/publicaciones/xml/0/27440/LC-W99.pdf [18-11-2007].
9. Surawski, A. and Cubillos, J., *Origen e Implementación del Programa Fondo Solidario de Vivienda*, Programa Ciudadanía, Participación y Políticas Públicas, INAP, Universidad de Chile, 2005.
10. Colin, B. ed., *International Public Debates: Urban Policies and The Right to the City*, Division of Social sciences Research and Policy, United Nations Educational, Scientific and Cultural Organization, Paris, 2006.
11. Ministry of Housing and Urbanism of Chile, *Ordenanza General de Urbanismo y Construcciones, Título 1: Disposiciones Generales*, Ministerio de Vivienda y Urbanismo de Chile, Santiago de Chile, 2007.
12. Kilian, A., Design Innovation through Constraint Modeling, in: Duarte, J. P., Ducla-Soares, G. and Sampaio, A. Z., eds., *Digital Design: The Quest for New Paradigms: Proceedings of the 23rd eCAADe Conference*, Technical University of Lisbon, 2005, 671-678.
13. Burrow, A. and Woodbury, R., Pi-Resolution in Design Space Exploration, in: Augenbroe, G. and Eastman, C., eds., *Computers in Building: Proceedings of the 8th CAAD Futures Conference*, Kluwer Academic Publishers, 1999, 291-308.
14. Gross, M., *Design as Exploring Constraints*, Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, 1986.
15. Tobin, K. L., Constraint-Based Three-Dimensional Modelling as a Design Tool, in: Goldman, G. and Zdepski, M. S., eds., *Reality and Virtual Reality: Proceedings of the ACADIA Conference*, Troy, 1991, 193-209.
16. Baykan, C. A. and Fox, M. S., Constraint Satisfaction Techniques for Spatial Planning, in: ten Hagen, P.J.W. and Veerkamp, P.J., eds., *Intelligent CAD Systems III: Practical Experience and Evaluation*, Springer-Verlag, Berlin, 1992, 187-204.

17. Medjdoub, B. and Yannou, B., Separating Topology and Geometry in Space Planning, *Computer Aided Design*, 2000, 32(1), 39-61.
18. de Vries, B., Jessurun, A.J., Kelleners, R.H.M.C., Using 3D Geometric Constraints in Architectural Design Support Systems, in: *WSCG 2000: Proceedings of the 8th International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media,* University of West Bohemia, 2000.
19. Eggink, D., Gross, M.D. and Do, E., Smart Objects: Constraints and Behaviors in a 3D Design Environment, in: Penttilä H. E., ed., *Architectural Information Management: Proceedings of the 19th eCAADe Conference*, Helsinki University of Technology, 2001, 460-465.
20. Russell, S. and Norvig, P., eds., *Artificial Intelligence: A Modern Approach*, Prentice Hall, New Jersey, 2002.
21. Peña, W., ed. and Parshall, S., *Problem Seeking: An Architectural Programming Primer*, AIA Press, New York, 2001.
22. Rich, E. and Knight, K., ed., *Artificial Intelligence*, McGraw-Hill Education, New York, 1991.
23. Minsky, M., *Heuristic Aspects of the Artificial Intelligence Problem*, Group Reports 34-55, MIT Lincoln Laboratory, 1956, op. cit. Alexander, C., ed., *Notes on the Synthesis of Form*, Harvard University Press, Cambridge, Massachusetts, 1967.
24. Kramer, G., A Geometric Constraint Engine, *Artificial Intelligence, Special Volume on Constraint-Based Reasoning*, 1992, 58(1-3), 327 – 360.
25. Michalek, J., *Interactive Layout Design Optimization*, Master of Science Thesis, University of Michigan, 2001.
26. Van Hentenryck, P., The OPL Optimization Programming Language. The MIT Press, Cambridge, Massachusetts, 1999.
27. Medjdoub, B., Richens, P. and Barnard, N., Building Services Standard Solutions: Variational Generation of Plant Room Layouts, in: de Vries, B., van Leeuwen, J. and Achten, H., eds., *Proceedings of the 9th CAAD Futures Conference,* Kluwer Academic Publishers, 2001, 479-493.
28. Loemker, T., *Plausibilität im Planungsprozess Umbau und Umnutzung als Optimierungsaufgabe*, Ph.D. Thesis, Bauhaus-Universität Weimar, Germany, 2006.

Dirk Donath
Bauhaus-University Weimar
Chair Computer Science in Architecture
Belvederer Allee 1, D-99 423 Weimar, Germany

caad@archit.uni-weimar.de

Luis Felipe González Böhme
Federico Santa María University of Technology
Department of Architecture
Avda. España 1680, Casilla 110-V, Valparaíso, Chile

luisfelipe.gonzalez@usm.cl