A Programming Language for Architectural Symbolic Modeling

*Alberto Paoluzzi** *Claudio Sansoni***

Abstract

In this paper a software project supporting architectural design is outlined. Such a project aims to develop the new design language PLASM (a Programming Language for Architectural Symbolic Modeling), which is planned to be a very high-level, user-oriented language, belonging to the class of constraint languages. The language PLASM will support a small set of abstract data types which are significant in various outstanding problems of architectural design, and will offer both procedural features and non-procedural constraints satisfaction. It will allow the designer to make use of a large set of computing tools in any phase of architectural design, in order to explore a wider set of design solutions. Customizable evaluation functions will be available in the language. The execution of a PLASM program may result either in generating or in updating a semantic network over a set of data objects solving the geometric problem under consideration. The proposed language will support both abstract data types significant in the design domain, and tools performing automatized data generation and transformations between different data types. The modification of any object in such a system, both performed by editing a daemon program and/or by interactively modifying a data object. with result in the immediate propagation of changes into the problem network, by activating a message passing mechanism.

*Universita' La Sapienza, Dip. di Informatica e Sistemistica, Via Buonarroti 12, 00185 - Roma (Italy), Fax n: (396)734616; E-mail: paoluzzi!i2unix!mcvax@disrm.uucp

**Universita' La Sapienza, Dip. di Progettazione Arch. e Urbana. Via A. Gramsci, 53, 00197 - Roma (Italy)

1 Introduction

The complexity of design problems is the main reason for introducing CAD techniques in the architectural design. Unfortunately, in spite of the progress in computing technology and of its widespread diffusion, the results of the computer application in architecture are largely disappointing. It is in fact widely recognized that CAAD techniques have had just a minimal impact upon the project quality, and therefore a minimal effect upon the building industry and the built environment. Our current design systems are, in most cases, just drafting tools, whose innovative aspects result from other design disciplines, like electronic and mechanical design, which primarily necessitate to efficiently organize the design work. For this reason, the best assisted design tools allow for structuring the design using layers and/or levels, simulating in this way an electronic circuitry or a mechanical assembly. Unfortunately, the actual problems arising in architectural designing are very different, although in producing a detailed design it can be sometime useful to use layers and levels. As a matter of fact, an architectural design is mainly developed through stepwise refinements, using a top-down strategy; at the contrary, electronic and mechanical design often require a bottom-up approach.

In spite of the large computer diffusion, many of the interesting ideas proposed in the seventies [Negroponte 1975, Mitchell 1979, Eastman 1975] had only a small influence on the professional practice, so that only a few systems can be considered able to-improve the architectural design. URBAN 5 [Negroponte 1972], developed by Negroponte at MIT between '70 and '75, can be considered the first CAD system oriented to architecture. Using URBAN 5 a designer could have an interactive dialog with the machine, and was able to experiment schematic design solutions by directly building a spatial model with 3D boxes. Pioneer examples of integrated CAAD for the actual utilization are the ARK-2 [Lee 1973] system developed by Perry Dean Partners Inc., and the OXSYS system [Richens 1974] developed by the Applied Research of Cambridge, in cooperation with Oxford Regional Health Authority, for the use with the Oxford Building System. In the academy, the ABACUS group at Strathclyde University experimented advanced procedures in order to evaluate alternative design solutions. Particularly interesting is the software GOAL [Sussock 1982] ('General Outline Appraisal of Layout') whose aim is to offer the designer evaluation tools which can be applied to the first design development phases. Other notable integrated systems are the Bijl's SSHA [Bijl 1974], for the housing design by using component based building systems, developed for the Scottish Special Housing Association by the Edinburgh University, and CEDAR (Computer-aided Environmental Design Analysis and Realisation) [Chalmers 1972], based on the building system SEAC, developed by a research group of the UK Department of Environment for the Post Office Telephone Engineering Centre.

The PLASM project, funded within the Finalized Research Program "Edilizia" (Building Technologies) of Italian National Research Council (CNR), aims to realize a 'Design Language' to be used also in the first, more creative, steps of the design process. Thee PLASM main goal is therefore to propose symbolic representation tools, suitable for architectural design, to be used in the context of a four generation language for the increase of

the designer productivity. Such a language will be oriented to the geometry description and will use a declarative approach. It will allow for generation and control of a whole set of geometry solutions, through the formalization of design constraint and objectives. The aim of this paper is to report about preliminary choices and preparatory work in the language development.

2 Object-Oriented and Constraint Based Languages

The design work is an iterative process whose solution space is delimited by a set of constraints and objectives [Simon 1969, Lawson 1980]. The designer proceeds euristically by reducing the freedom degrees of the design hypothesis, until the possible variations are reduced to zero. This means that the design is defined incrementally, by iteratively adding new constraints, and by continuously verifying the partial results against the objectives. In the following of this section we shortly recall the main aspects of object-oriented programming languages and of languages based on constraint satisfaction. We believe that the new design language PLASM should belong to both these language classes.

In procedural programming languages it is usually necessary to distinguish between procedures and data, which are treated in a separate way. In object-oriented programming data and procedures are instead combined into units called *objects*. In such languages an object is defined as a container of some data type and of the procedures which are necessary to operate over such data. With such approach it is no longer necessary to communicate to the outside world the details of the implementation of the operations. It is furthermore promoted the modularity of the computing environment, which appears particularly useful when the complexity of the system increases. Object-oriented programming languages can be considered derived by the *abstract data type* concept, derived in the context of the theoretical computer science [Aho et al., 1983]. An abstract data type is defined as a pair (A, O(A)) constituted by a data set A and by a set of operators O(A) over such data. In object-oriented languages is often done a distinction between data *classes* and *instances*. A data class defines a particular type of object; a data instance is a particular copy of an object in a specified class. Usually the class contains *the behaviour,* i.e. the procedures which apply to all the objects into the class; an instance contains the data values concerning a particular exemplary of the considered object. Classes can often be considered as *subclasses* of other classes *(superclasses)* of which they hereditate the behaviour. Normally each class is created at the beginning of a program, whereas the instances are created during a work session.

In traditional programming languages, the programming follows an imperative style, where the programmer must describe, step by step, the behaviour of the program in any possible situation. With such approach the design and description of an algorithm tend to cover and to hide the problem resolution strategy. Besides, the programming effort is often so high that a non-specialist user is discouragiate to write his own program. At the contrary, in programming languages based on *constraint satisfaction* the programming is

declarative: the programmer has to specify a set of relations between a set of objects, and is the system job [1] to find one or more solutions which satisfy such set of relations. In such a way the user is not requested to specify an algorithm solving the problem, but just to describe the problem itself. The possibility of solving many different problems (in a given domain) is an essential characteristic of constraint languages. Leler [Leler, 1988] summarizes in the following way the main aspects of constraint based programming: a *constraint* expresses a desired relationship between two or more objects; a *constraint based language* is a language used to describe both objects and constraints; a *constraint based program is* a program written in a constraint based language. It is utilized to describe a particular set of objects and a particular set of constraints; a *constraint satisfaction system* finds solutions for constraint based programs, by using problem solving methods, often oriented towards a particular problem solving domain. In the next section we will describe some particular classes of geometric objects, that we believe useful in describing the geometric behaviour of an architectural design.

3 PLASM Objects

Forrest wrote in a fundamental paper [Forrest, 1974] : '... we have had considerable success with a relatively simple command interpreter which has been used to implement a wide variety of problem oriented design and command languages. However such languages do not seem to have the same glamorous appeal as algorithmic languages and their design and implementation has been rather neglected. Once more, working in this area would be of great practical value.'

Fifteen years later, design languages in the sense of Forrest have found little diffusion. Nevertheless, a symbolic description is able to describe the whole process of shape generation, and not only the final result. In the PLASM project we aim to develop a design language in the sense of Forrest, able to specify symbolic design representations, as well as to represent design parts, and processes and operations used in shape generation, at various detail levels. Consequently, the language must allow both for the incremental design development and for automatic recomputing of all data, as consequence of any editing of part or constraint description. In order to do this, the PLASM language must permit the definition of abstract data types significant for architectural designing, and must provide both procedural control structures and non procedural mechanisms of constraint satisfaction. The objective of a work session will be that of generating a whole set of geometrical solutions for a design problem. In particular, the designer, by using appropriate shape descriptors and operators, must be able to generate and to explore a range of solutions in order to accomplish choices based on comparative evaluation of alternatives. Furthermore, the designer must be free of utilizing previously obtained partial solutions, and of defining new ones by editing the symbolic descriptions already available.

We like to refer as a precursor of our approach the famous SKETCHPAD system by

[1] By using some suitable problem solving technique.

Ivan Sutherland [Sutherland 1963], which first introduced a number of important concepts in the area of interactive graphics and design modeling. Another outstanding reference is THINGLAB [Borning 1979] by Alan Borning, developed with Smalltalk at Xerox Labs, for modeling simulation experiments in the domain of geometry and physics. Other direct references are KRL by Bobrow e Winograd, used as knowledge representation language, and CONSTRAINTS [Stallman and Sussman1977; Steele and Sussman 1980] 'a language for expressing almost-hierarchical descriptions', mainly used in the domain of electrical network analysis. The more significant reference for the PLASM project will probably be the general purpose specification language BERTRAND [Leler 1988], which permit the user to specify problem -specific objects and constraints, and performs constraint satisfaction by using data typing and generalized rewriting rules.

3.1 Relation Graphs

The high-level structural organization of an architectural design can be defined without reference to any specific decision about shape. For example, both the activities to be performed inside the planned building and the required spaces can be described by using relation graphs [Jones, 1970] (fig.2). Such kind of representation does not determine the shape of the project, but nevertheless strongly constraints its structure. In particular, by using graphs it is possible to readily specify various binary relationships among spaces, concerning the required adjacency, accessibility, etc. All kind of binary relationship can be requested without assuming any specific decision about shape.

Among relational operations having a major interest in the design process we recall: (a) adding (deleting) a node (arc). (b) expansion (contraction) of a node (subgraph) into a subgraph (node). The first operation is used to create or eliminate an activity or space, with the consequence of reorganizing the whole set of requested relationships; the second operation concerns the possibility of representing hierarchically the most complex relationships in a project. It is well known, for instance, that a dwelling can be considered, at a proper abstraction level, as a unique graph node, and at another abstraction level as a subgraph having as nodes its constituting spaces [see, e.g., Carrara and Paoluzzi, 1981].

In architectural CAD we are mainly interested with operations over weighted undirected graphs, in order to evaluate clusters, flows and distances. Some graph operations and functions having design interest are the following: (a) section [2] having minimal (maximal) flow; (b) minimal distance between a node and all the remaining nodes; (c) chain of minimal length between any pair of assigned nodes.

2 Minimal subset of arcs which disconnects the graph.

3.2 Topological Graphs

When the binary relationships between the design components are expressed with the aim of defining some shape organization, they can be modeled by using topological graphs (fig.3). In order to define a topological graph it is firstly necessary to introduce the concept of *planar graph*. *A* graph is said *planar* when it can be draw on the-plane without crossing of edges (arcs) outside of vertices (nodes). A planar graph can be represented in many equivalent ways on the plane. Two drawings are equivalent is they can be bicontinously transformed each into the other. In other words two planar graph representations are equivalent if they differ just for the vertex coordinates and for the shape of edges, but they split the plane in the same set of connected regions, which are called faces. An equivalence class of planar representations of a given graph is said *topological graph*.

A graph can be assumed as a model of the shape topology only if it is planar. In the opposite case it is not possible to embed the graph in the plane. The planarity test [Hopcroft, Tarjan 1973] assumes the greatest importance in the context of design by constraint. Another important graph operation is that which associates a topological graph together with its *geometrical dual* [Harary 1969]. Such operation allows to associate an *adjacency graph* with its dual *plan graph* and vice-versa [Steadman 1983]. The more important operation concerning duality is that which associates a planar graph with the set of the maximal planar graphs containing the given graph as a subgraph. The constructive combinatorial algorithm has been given by Cocomello and Paoluzzi in [Cocomello, Paoluzzi, 1981]. Unfortunately, such set of maximal planar graphs is "p-space hard", and therefore can be actually constructed only for graphs of cardinality $n \leq 10$. For each maximal planar planar graph the corresponding dual plan is unique and can be constructed in a very direct way. If an adjacency graph represent a set of requested adjacencies for a set of spaces, anyone of the plan graphs associated with a maximal planar graph (containing the adjacency graph) represent one different feasible topology for a plan which satisfies all the requested adjacencies. Also the inverse transformation is very interesting: the common (dual) subgraph shared by many plan graphs represents the relational constraints which are simultaneously satisfied by all the considered plans.

3.3 Polygonal Structures

If the topology and the geometry of design are defined completely and explicitly, we call the corresponding object *polygonal* graph, whose faces are polygons. In this case we call *arcs* the polylines shared by two adjacent faces, and *nodes the* crossing of at least three arcs. The more important unary operation applicable to a polygonal- graph is the *affine transformation,* used to apply a sequence of geometric transformations (translation, scaling, mirroring and rotation) to a polygonal graph. A set of polygonal graph can be joint to constitute a polygonal *structure,* where polygons are connected by affine transformations, using the hierarchical modeling technique well known in graphics [see PHIGS, 1985]. Affine transformations over polygonal structures are very useful because they work over "voids" with-

out affecting their "boundaries", which are not yet specified in this kind of representation (fig.4). The main property of such a representation is evident: a polygon structure can freely manipulate the spaces, without to deform the internal partitions or the external envelopes, because they are defined only in a virtual way. Over the polygonal representation (graphs and structures) can be defined the following binary composition operations: (a) join; (b) union; (c) intersection; (d) difference; (e) disjoint sum. Other useful (external) operations, which transform 2D polygonal structures into 3D structures are the following: (f) rotational sweeping; (g) extrusion; (h) interpolation; (i) xor-sweeping; (l) or-sweeping.

Let us notice that any traditional design technique makes large use of 2D orthographic projections. The set of orthographic projections and of complementary geometric information needed to specify completely a 3D object is called a 2.5D *representation*. 2.5D reprs are frequently used in architectural design, because buildings are described by sections, which are constant "almost everywhere". This characteristic makes 2.5D representation, closely associated with sweeping operation, particularly useful for CAAD, because it allows to generate an informationally complete solid model starting from a finite set of 2D sections. In particular, a 3D polygonal structure can be readily obtained by sweeping a 2D polygonal structure, where each cycle is associated with some suitable "heigth" information. In fact each cycle can individually generate a 3D solid cylinder, subsequently united or glued together to obtain the complete 3D model.

We have said that a polygonal structure, both 2D and 3D, is a set of geometrical elements without width. Such schematic descriptions of a building object can be utilized, together with a schematic description of building "walls", to generate a solid (in the following called "polyhedral") description of the building. Such schematic description of the geometry of building fabric will be called in the following *wall representation* (fig.6). A wall repr. contains both a set of simple rules to associate widths to the space boundary elements, and priorities to be applied in case of conflicts in the occupance of space. A wall representation is therefore defined as a data structure containing: (a) a taxonomy of functional subsystems (vertical enclosures, internal partitions, horizontal enclosures, etc); (b) some geometrical and/or topological rules which associate functional subsystems to the geometrical elements of a polygon structure; (c) some sweep rules necessary to associate widths to the geometrical elements; (d) some combinatorial rules which partitionate the building subsystems in building components [Mandolesi, 1982]; (c) a subsystem ordering, to be applied in solving space conflicts in automatic generation of detailed building solid model. For example, if two solid "walls" are in conflict, i.e. if they occupy the same space portion, one of them should be substituted by its difference with the other solid.

3.4 Variational Representation

Sometimes it is necessary to modify previous design decision, where such changes have to extend to all the connected design entities. An useful approach is constituted by parametric representations, especially diffused in mechanical design. A parametric representation, called 'primitive instancing' by A. Requicha [Requicha, 1980], is a procedure which is

able to generate a complex shape depending on the instantiation of some shape parameters. When the relationships between parameters are well formalized and can be expressed as a set of simultaneous equations, being any solution computable by solving the system of equations, they are referred to as *variational geometry* [Light and Gossard, 1983; Gossard et alt., 1988].

With the aim of introducing a valiational approach in PLASM, we give some definitions. A *geometric relation* is an identity which associates, by means of relational operators ($<, \leq, =, \geq, >$), two algebraic expressions between the vertex coordinates of a polygonal structure. Such algebraic expressions can be associated to symbolic identifiers; e.g., for the polygon shown in figure 7 we can write both the following system of algebraic equations

$$side_1 = X_2 - X_1 \quad (1)$$
$$side_2 = Z_3 - X_4 \quad (2)$$
$$side_3 = X_5 - X_6 \quad (3)$$
$$side_1 = side_2 + side_3 \quad (4)$$

and the unique equation in six variables obtained by substitution:

$$(X_2 - X_1) = (X_3 - Z_4) + (X_5 - X_6). \quad (5)$$

In conclusion, a variational representation is a set of simultaneous shape constraints, where both the topology and some shape invariants are fixed. This approach has been described mainly by [Light and Gossard, 1983] in the context of mechanical CAD. [Paoluzzi, 1987] and [Cattani and Paoluzzi, 1988] have shown how to express integral constraints (volume, centroid, moments) concerning parametric polyhedra: they can be written as polynomial equations in the shape parameters. By using the variational approach the design dimensioning is obtained as the solution of a set of simultaneous equations where each constraint is a polynomid equation, non linear in the general case. By solving such a system, usually linked to the topology of a shape configuration, it is possible to obtain the set of feasible geometries for the considered design problem. Such an approach is well known to the architects, as far as from the antiquity (Figure 8) [Chitham 1987]. The set of rules establishing dimensional relationships among the parts of the ancient Greek temple are an eloquent example. In the contemporary age Le Corbusier's Modulor [Le Corbusier 1965] demonstrated that by imposing particular ratios between part dimensions it is possible to obtain an harmonic control of shape. Lansdown [Lansdown, 1987] in a recent paper has proved how the parametric approach is stimulant for the architectural design.

Optimization techniques are another useful tools to explore dimensional variations of shape. When a set of dimensioning constraints has been established, it is in fact possible to explore feasible solutions with the aim of one or more objective functions. The more common and easily applicable technique of mathematical optimization is the so-called Linear Programming. In actual cases may be very difficult to express geometric dimensioning as a linear programming problem: often, e.g., geometrical constraints concern surfaces or volumes, or geometric properties like orthogonality, which cannot be linearly expressed.

Anyway, in the past decade, various applications of optimization techniques in plan dimensioning have been proposed. Mitchell, Steadman and Liggett [Mitchell et al., 1976] proposed linear programming to set optimal dimensions of a mobil-home. Gero and Radford [Radford and Gero, 1988] demonstrated the use of non-linear programming for the same problem. More recently Balachandran and Gero [Balachandran and Gero l987] showed the solution of an example with three conflicting objectives. Another line of thought, considering a geometric plan as the solution of a LP program minimizing the number of angles in a orthogonal polygon arrangement, originated by the work of [Carrara, Gori-Giorgi and Paoluzzi, 19771 and was fully exploited by R. Tamassia [Tamassia, 1988] for the automatic drawing of CASE schemata.

3.5 Polyhedral Structures

Polyhedral models can be used as natural and *complete* representations of architectural objects. In fact a set of polyhedra allows for contemporary and completely modeling the geometry of both full and void spaces in a building. In PLASM language a *polyhedral structure* will be defined as a container of polyhedral data structures and of references to external polyhedra, in a way similar to that defined in PHIGS for graphical structures. Each individual polyhedron will be represented by using the winged-triangle data structure defined in the solid modeler *Minerva* developed in recent years at the University of Rome "La Sapienza" [Paoluzzi and Masia, 1989]. Such a representation allows to represent a boolean algebra over linear polyhedra, where each model may be also unconnected and may have any topological degree. A winged-triangle representation is furthermore stored in a quantity of computer memory less than that used by other types of boundary representations [Baumgart, 1972; Braid, 1980; Woo, 1985]. The more important operations over polyhedra are the regularized set operations (intersection, union, difference) [Paoluzzi, Ramella and Santarelli, 1989] and the integration of monomials over polyhedral domains [Cattani and Paoluzzi, 1989].

References

[1] A. V Aho, J. E. Hopcroft and J. D. Ullman. *Data Structures and Algorithm*. Addison-Wesley, Reading, MA,1983.

[2] D. Balachandran, J. S. Gero. Dimensioning of architectural floor plans under conflicting objectives. *Environment and Planning B,* 14 (l):29-37, July1987.

[3] B. G. Baumgart. *Winged-Edge Polyhedron Representation*. Computer Science Department, Report n. CS-320, Stanford University, 1972.

[4] A. Bijl. Research in progress: Edinburg CAAD studies. *Computer Aided Design*, 6 (3): 183-186, March1974.

[5] 1. C. Braid. *Notes on a Geometric Modeler*. CAD Group Document n.101, University of Cambridge, UK, 1979 (Revised 1980).

[6] G Carrara and A. Paoluzzi. A *System Approach in Building Component Design*. in "System Approach for Development, M.A.R. Ghonaimy (ed.), Pergamon Press, New York, 1979.

[7] G Carrara C. Gori-Giorgi and A. Paoluzzi. A *System Approach to Building Program Planning*. CABD Lab RR 80.02, Ist. di Arch., Edilizia c Tecn. Urb., Univ. "La Sapienza", Rome, 1980.

[8] C. Cattani and A. Paoluzzi. Boundary Integration over Linear Polyhedra. to appear in *Computer Aided Design,* December, 1989.

[9] J. Chalmers. *The development of CEDAR. Proc*. of the Int. Conf. on *Computers in Architecture,*York, British Computer Society, London,1972.

[10] R. Chitham. *The classical orders of architecture*. Architectural Press Ltd., London, 1985.

[11] A. Borning. *Thing-lab A contraint-oriented simulation laboratory* Technical Report, Xerox Parc Palo Alto, California, 1979.

[12] C.M. Eastman. *Spatial Syntheses in Computer-Aided Building Design*. Applied Science, London, 1975.

[13] A.R. Forrest. "Computational Geometry - Achievements and Problems". In Com*puter Aided Geometric Design,* R.E. Barnhill and R.E Riesenfeld (eds.), Academic Press, New York, 1974.

[14] D. C. Gossard, R. P. Zuffante and H. Sakurai. Representing Dimensions, Tolerances, and Features in MCAE Systems. *IEEE Computer Graphics and Applications,* 8(2):5 159, March 1988.

[15] E Harary. *Graph Theory*. Addison Wesley, Reading, Massachusetts, 1969.

[16] J.E.H. Hopcroft, R. Tarjan. Efficient Planarity Testing. *Communication of the ACM,* July 1977.

[17] J.C. Jones. *Design Methods*. John Wiley & sons, New York,1970.

[18] J. Lansdown. "Computer Graphics in Design: Parametric Variation as Design Method", in D.F.Rogers, R.A.Earnshaw (eds.), *Techniques for computer Graphics*. Springer Verlag, New York, 1987.

[19] B. Lawson. *How designers think*. The Architectural Press Ltd., London 1980.

[20] Le Corbusier *Le Modulor*. Editions de l'Architecture d'Aujourd'hui, Boulogne, 1965.

[21] K. Lee. *Computer Aided Architectural Design: 16 ARK-2 Articles, Environmental Design and Research*. Technical Report, Environmental Design and Research Center, Boston, Mass., 1973.

[22] Wm. Leler. *Constraint Programming Languages*. Addison -Wesley, Reading, Mass.,1988.

[23] R. Light and D. Gossard. Modification of Geometric Models through Variational Geometry. *Computer Aided Design,* 14(4)209-214, July 1982.

[24] N. Negroponte. *The Architecture Machine*. The MIT Press, Cambridge,1972.

[25] N. Negroponte. *Soft Architecture Machine*. The MIT Press , Cambridge, 1975.

[26] E. Mandolesi. *Edilizia*. *U.T.E.T.,* Torino, 1982.

[27] W. Mitchell. *Computer Aided Architectural Design*. Van Nostrand Reinhold, New York, 1979.

[28] WJ. Mitchell, J.P Steadman and R.S. Ligget. Synthesis and optimization of small rectangular floor plans. *Environmental and Planning B,* 3(l): 37-70, January 1976.

[29] A. Paoluzzi. *Integration Constraints in Parametric Design of Physical Objects*. Technical Report TR 87-804, Department of Computer Science, Cornell University, Ithaca, NY, January 1987.

[30] A. Paoluzzi and M. Masia. The Geometric Modeler Minerva. *Wheels for the Mind,* Apple Europe, (2)14-32, April 1989.

[31] A. Paoluzzi, M. Ramella and A. Santarelli. A Boolean Algebra over Linear Polyhedra. to appear in *Computer Aided Design,* September, 1989.

[32] A.D. Radford and J.S. Gero. *Design by optimization in Architecture building and construction*. Van Nostrand Reinhold, New York, 1988.

[33] A. A. G. Requicha. "Representations for Rigid Solids: Theory, Methods and Systems". *ACM Computing Surveys,* 12(4):437-464,1980.

[34] P. Richens. *OXSYS-Computer Aided Building for Oxford Method*. CAD74, Proceeding of the Conference on Computer Aided Design , London, 1974.

[35] H.A. Simon. *The science of artificial*. MIT Press, Cambridge, 1969.

[36] J.P. Steadman. *Architectural Morphology*. Pion Ltd., London, 1983.

[37] H. Sussock. *Goal* : user *manual 2.7*. Abacus occasional paper n.62, 1982.

[38] R.M. Stallman and J. Sussman. Forward referencing and dependency-directed backtracking in a system for computer-aided circuit analysis. *A J. journal,* Sept. 1977, 135-196.

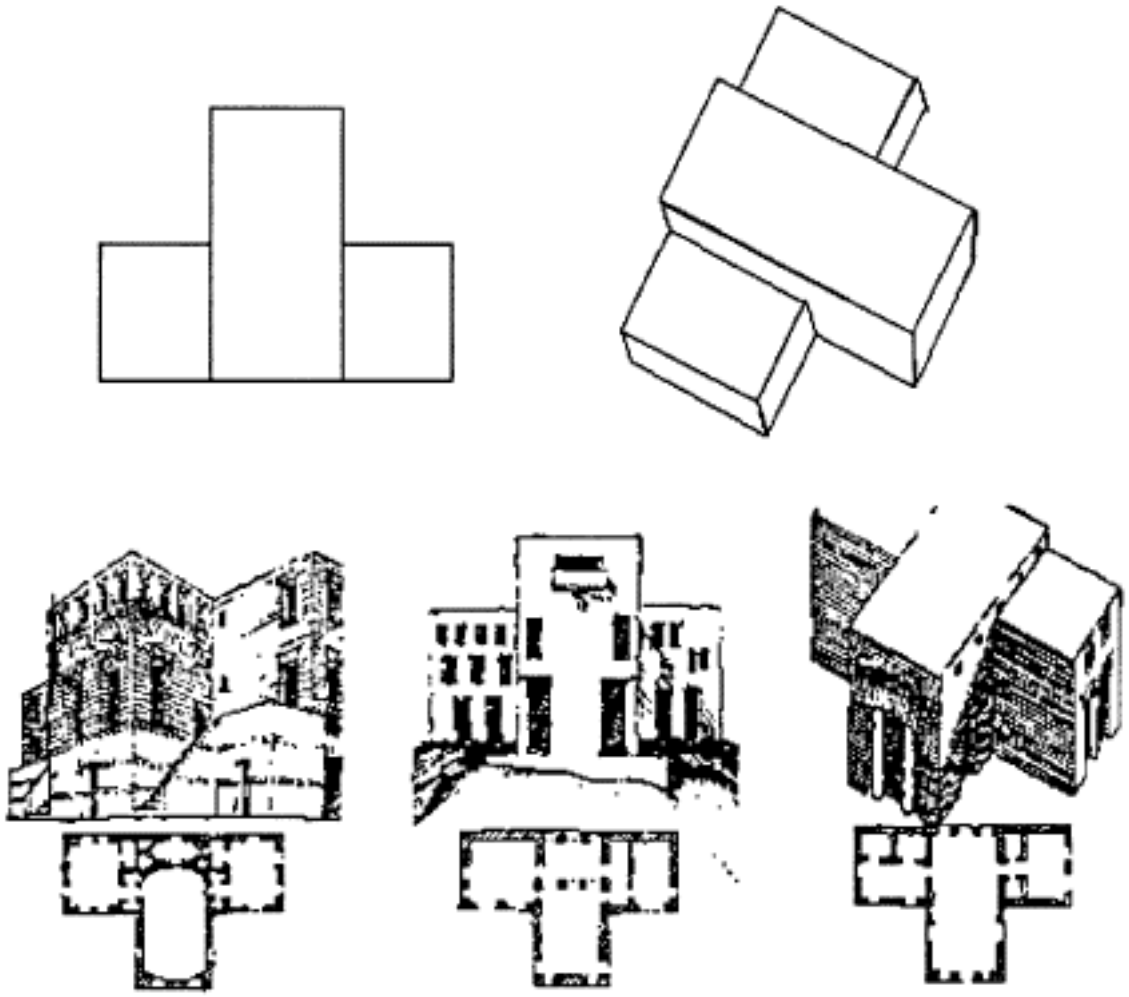

Figure 1. A 'bottom up' approach in architectural design (drawings of R. Krier)

[39] G. Steele and J. Sussman. Constrains a language for expressing almost hierarchical descriptions. *A I. journal,* Aug. 1980, 1-40.

[40] I. Sutherland *SKETCHPAD: A man-machine Graphical Communication System*. IFIPS Spring Joint Computer conference*, 1963*.

[41] T. C. Woo. "A Combinatorial Analysis of Boundary Data Structure Schemata". *IEEE Computer Graphics and Applications*, (3): 19-28, 1985.

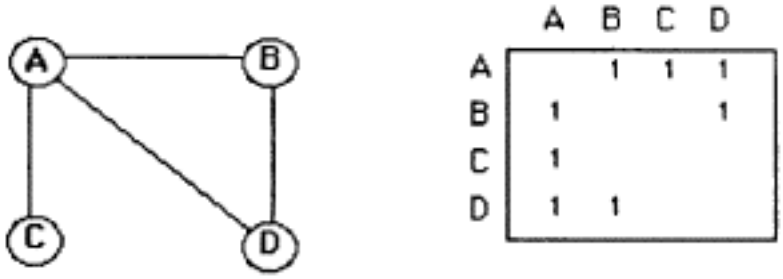

Figure 2. Relation graph.

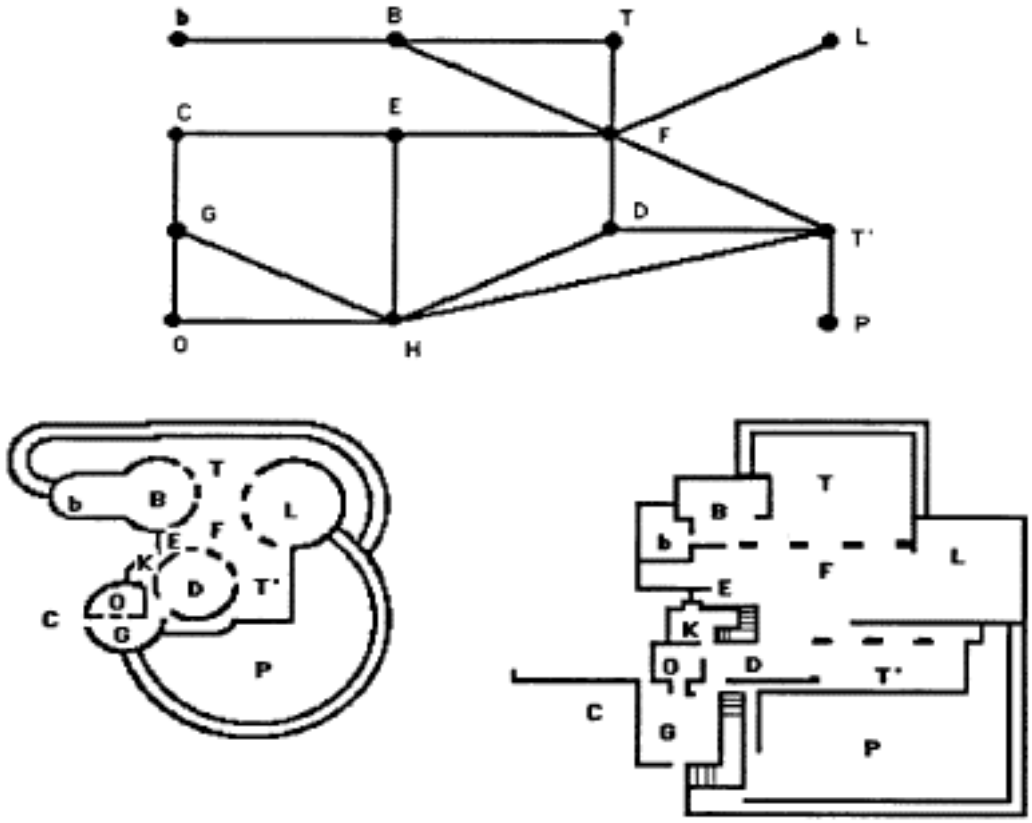

Figure 3. Topological graph

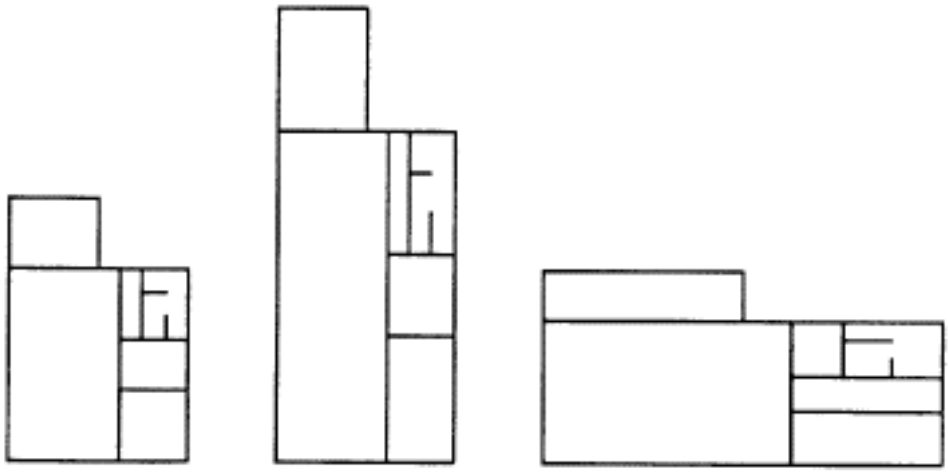

Figure 4. Affine transformations over polygonal graphs

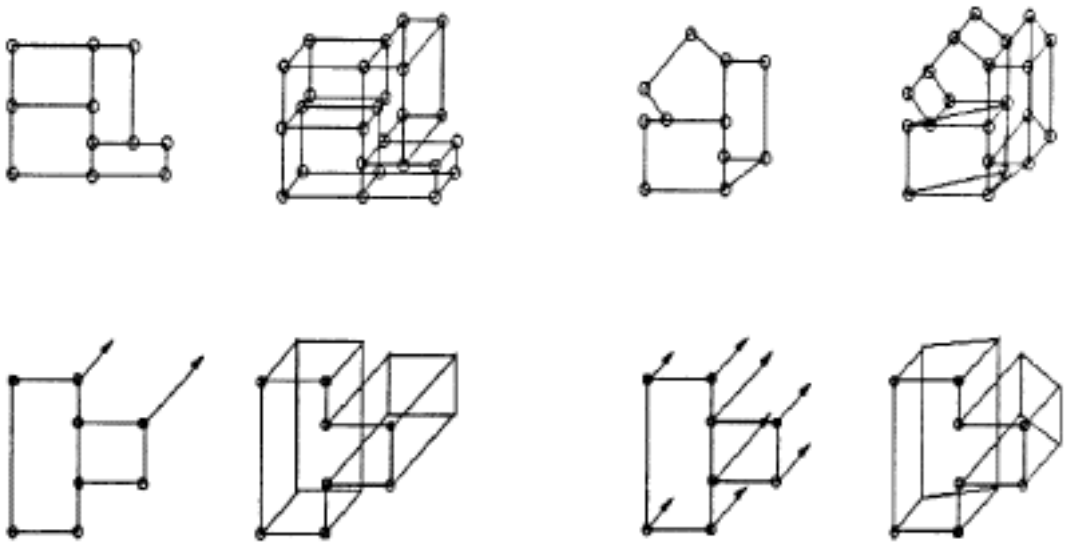

Figure 5. 3D structures generation by sweeping of 2D structures.

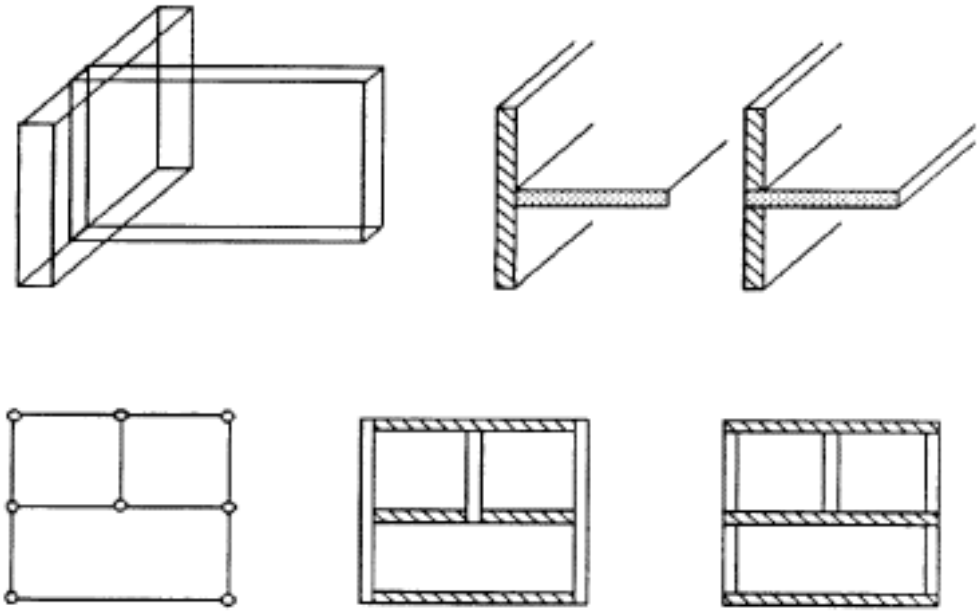

Figure 6. Wall representation

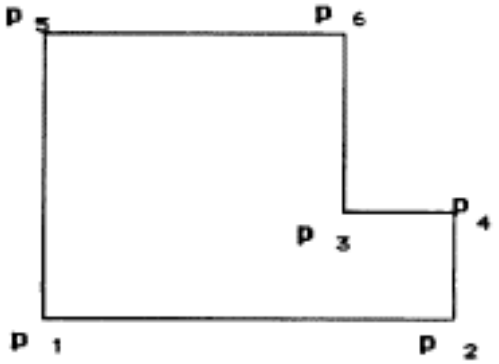

Figure 7. Variational representation

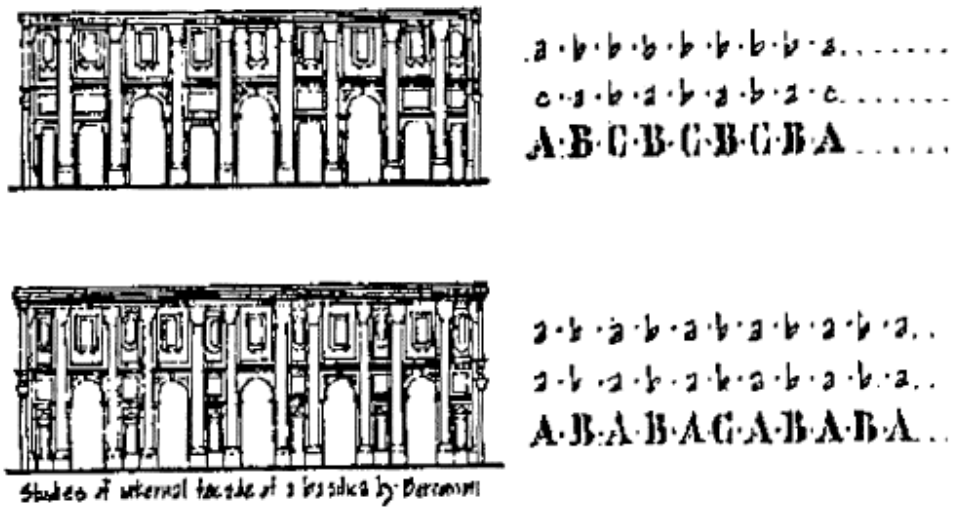

Figure 8. Some prospect rules (Borromini)

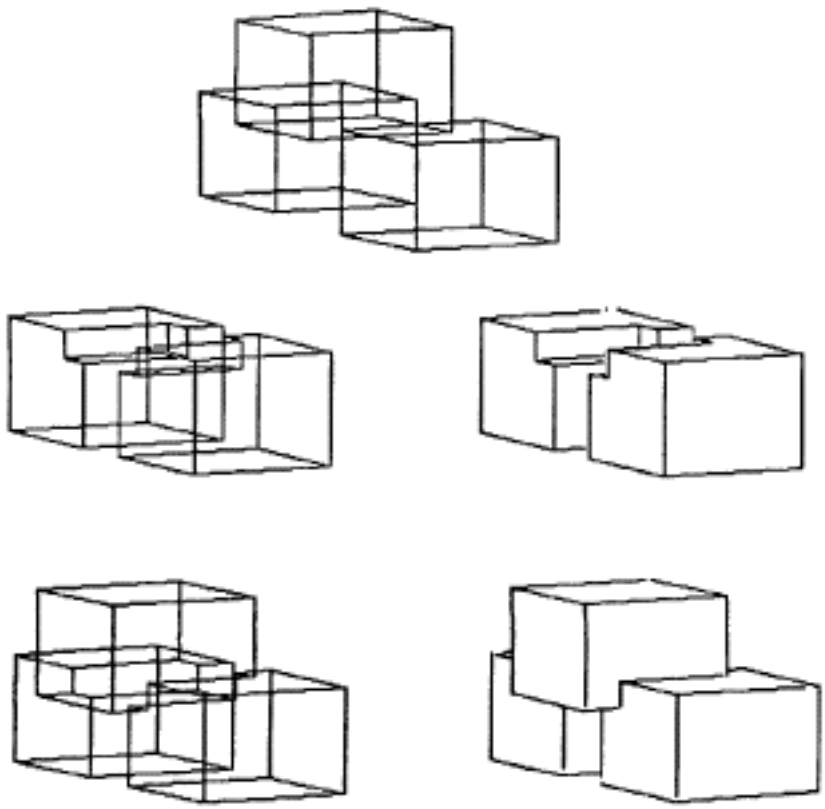

Figure 9. Set Operations over polyhedra (Minerva modeler)