ELSEVIER

Automation in Construction 9 (2000) 369–377

AUTOMATION IN CONSTRUCTION

www.elsevier.com/locate/autcon

Parametric design: a review and some experiences

Javier Monedero [*]

Departamento de Expresión Gráfica Arquitectónica, Universitat Politecnica de Catalunya, Diagonal 649, 08028 Barcelona, Spain

Abstract

During the last few years there has been an extraordinary development of computer-aided tools intended to present or communicate the results of architectural projects. But there has not been a comparable progress in the development of tools intended to assist design to generate architectural forms in an easy and interactive way. Even worse, architects who use the powerful means provided by computers as a direct tool to create architectural forms are still an exception. Architecture continues to be produced by traditional means using the computer as little more than a drafting tool. The main reasons that may explain this situation can be identified rather easily, although there will be significant differences of opinion. In my opinion, it is a mistake trying to advance too rapidly and, for instance, proposing integrated design methods using expert systems and artificial intelligence while no adequate tools to generate and modify simple 3D-models are available. The modeling tools we have at the present moment are unsatisfactory. Their principal limitation is the lack of appropriate instruments to modify interactively the model once it has been created. This is a fundamental aspect in any design activity, where the designer is constantly going forward and backwards, re-elaborating once and again some particular aspect of the model, or its general layout, or even coming back to a previous solution that had been temporarily abandoned. This paper presents a general summary of the actual situation and recent developments that may be incorporated to architectural design tools in a near future, together with some critical remarks about their relevance to architecture. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Geometric modeling; Architectural and building models; Parametric design

1. Current 3D-models

In architecture, 3D-models are elaborated by some commercial version of one of the following techniques: polygonal meshes, solid models or parametric surfaces such as nurbs. Most architectural models are still produced using the first method, together with some appropriate interface that allows the use of commands such as ‘‘3dfaces’’, polylines with ‘‘width and thickness’’ or ‘‘revsurfs’’, ‘‘tabsurfs’’, ‘‘rulesurfs’’, etc. This is due to the characteristics of architectural models that are mainly composed of planar surfaces. Many architects work with what can still can be called 2.5D-models (wide lines or polylines depicting walls extruded to a particular height) that can be used both as drawing planes and simple 3D-models. Solid models are also widely used due to the fact that they allow boolean operations to create more complex forms. Nurbs or the like are rarely used (except by Frank Gehry), as common budgets do not favor sculptured or free-form surfaces. The history of 3D geometric modeling is studied and can be found in well-known computer books like Foley’s general exposition of computer graphics or Mortenson’s more specialised textbook on geometric model-

[*] Corresponding author. E-mail: javier.monedero@egal.upc.es

ing. This justifies a very shortened summary. The intention of this summary is not only to locate the subject in the adequate context but also to stress the distance in time that has separated a published paper and a generally used technique. As we will see, this distance is approximately the same that separates the first published papers on parametric design from our immediate future, let us say 2 or 4 years. That is to say, the situation is mature for a change in the current techniques used in CAAD; it has already happened in CAD/CAM although most of the architects that work with computers are unaware of it.

1.1. Evolution and limitations of CAD modeling tools

The first methods and techniques were put into practice during the 1960s and included basic 2D-primitives, as well as new entities like splines. The work of Bezier and De Casteljau goes back to this period. This was extended to 3D wireframes and surfaces patches. New graphical methods are to be associated to the name of Sutherland and to the year 1963, in which his thesis was published. Polygonal meshes were used at the end of the 1960s; soon there were techniques to visualize them by such methods as what is now currently known as flat-shading (Bouknight, 1970) or, better, Gouraud shading (1971) or, even better, Phong shading (1975). This is where most systems stop nowadays (more than 22 years thereafter!). Free-form or sculptured surfaces were developed extensively during the 1970s. The most advanced method currently used, nurbs (non-uniform rational b-splines) can be traced back to an article by A.R. Forrest in 1980. AutoCad has incorporated it, a couple of years ago (15 years after it was introduced), through an extra module (AutoSurf) that came together with version 13.

Solid modeling using a primitive form of CSG was also born at the beginning of the 1960s (at the MAGI labs in USA), and evolved rather slowly until some complete products appeared in Europe and USA at the beginning of the 1970s. The first important commercial packages, like Romulus, commercialized by Evans and Sutherland in 1980, appeared at the end of the 1970s. A very much quoted article by Requicha that summarized the state of the art at that time, with five principal systems quoted was also published in 1980. At the present time, most systems currently used combine two systems, B-Reps and CSG or use B-Reps as a shell that allows multiple representations and favors the transit from one to the other. This is the case with ACIS (Alan, Charles and Ian System) used by AutoCad since version 13 after dropping AME, a CSG system that did not work as it should.

1.2. Editability of current 3D representations

All these systems suffer from severe limitations from the point-of-view of an interactive approach to design:

- Lack of resources to edit surfaces. This is particularly clear in the case of sites that must be reworked and adjusted to receive a building.
- Lack of resources to edit volumes in a really interactive way.
- Lack of resources to maintain relations between parts of a volume during modifications.
- Lack of integration between surfaces and solids.

In the CAD/CAM-community, the methods currently used to create 3D architectural models have been abandoned a long time ago. Different types of research have been carried out to improve the situation. We offer some hints before getting into parametric design.

1.3. Object-oriented 3D-model: E-Reps

With a solid model such as the kind currently used in the architectural community, if one wishes to modify, for instance, a hole in a wall, one has to edit the csg tree, locate the primitive and then order the system to rebuild the tree. With an object-oriented approach, interaction will be more convenient and easier to manage. The internal data structure and the implementation of the algorithms able to modify it, are hidden in the object. In this way, the orders sent to the object do not need to specify how a modification shall be done but only what is to be done (for instance, change the position of a hole in a wall). The mechanisms of inheritance that relate classes to superclasses or subclasses assure that the previously specified relations will be maintained. Unfortunately, this requires an internal representation of data that is still lacking in current CAD systems. In general, a representation that uses a chain of references that links entities in the model is called a model graph. Hoffman [8] has introduced the term of E-Rep (editable representation) to denote this kind of structure. This has implications being of great interest in the future. This structure is similar to a CSG graph but with some important differences. The leaf nodes of the graph, in the CSG are usually the lowest primitives of the system, i.e., half spaces, whereas in the E-Reps, there usually are B-Reps. Also, in a CSG, the nodes are a few operators, mainly boolean operators, while in an E-Rep, the nodes may represent a wide range of types including sketches, sweeps, feature attachments, blends or dimensions. On the other hand, the CSG-graphs have a well-defined semantics and a guaranteed validity. This is not the case with E-Reps so it seems that there is still some experimentation needed.

2. Parametric design

Parametric design is, in a sense, a rather restricted term; it implies the use of parameters to define a form when what is actually in play is the use of relations. I will use the term in a wide sense that covers what can be found in the literature under other headings such as relational modeling or variational design or constraint-based design or other titles that will be quoted to some extent in the following paragraphs.

It should also be noted that, from an elementary point-of-view, there is not a clear boundary between what can be called parametric design and what is currently called computer-aided drafting or modeling. In these cases, forms are created by combining basic entities that are inserted in the model after a basic template, which includes their ‘‘proper parameters’’, is filled. A line, for example, is an entity that becomes part of a model once two parameters, its length and its direction, are specified. A polyline is a set of lines joined at their vertices whose position parameters must also be specified when it is created. A prismatic meshed volume is inserted in a model through four parameters, its location, length, width, and height. Besides this, we can also define ‘‘blocks’’ (AutoCad), ‘‘cells’’ (Microstation) ‘‘symbols’’ or ‘‘components’’ (other systems) that combine and keep together these primitive forms with different overall values. There are also, in current CAD systems, tools that allow us to make some modifications a posteriori regarding these primitive entities. However, this does not work for complex elements where we want relations to be maintained while modifying their parts independently. We can define a metal window as a block but if we change the scale at the moment of insertion, frame sections will change in the same proportion as the overall magnitude and we will not be able to keep a standard frame with different opening dimensions. But we can still define a procedure, through some programming language like AutoLisp, in such a way that only the relations are specified and the adequate dimensions are defined only at the moment of insertion in the model. This is already parametric design in a literal and fundamental sense. And, it is obviously of interest in the case of architecture due to the fact that a very important number of building elements can be grouped in families that tend spontaneously to be parameterized. And, if this can be done in a satisfactory way, it can save a lot of time and computer memory and will also help the management of these elements. As the notion of family is important in a parametric design we can define it formally: a set of elements that only differ in the dimension of their parts. To describe a family, to elaborate a primary design of a family, we only need two things: a topological description specifying the parts that constitute it and the relations they maintain with each other and a dimensional scheme specifying priorities and dimensional constraints. In this way, we can define an abstract collection of elements and insert them in our models. This is good for a start, but what happens if after the element is inserted we want to modify it? This is where parametric design, in a promising way, properly started out in CAD/CAM a few years ago in relation with the fundamental notion of constraint.

3. Constraints

A fundamental problem in CAD is how to make explicit some intuitive knowledge we have about something in such a way that a machine can interpret and treat it in an automatic way. This problem reveals its magnitude as soon as we try to formulate what is comfortably referred to as ‘‘common sense’’. From an architectural point-of-view, this is like knowing that floors ‘‘shall always be’’ horizontal or that windows ‘‘belong to’’ a wall and trying to formulate this knowledge in such a way that a machine could not violate such an obvious rule. This is dealt with by means of constraints. Constraints appeared in CAD as early as 1963, in the pioneer work of Sutherland. As it happens with the very notion of parametric design, the notion of constraint is present, in a basic way, in any CAD system. A polyline, for example, can be understood as a collection of curves with vertices constrained to remain attached. But, in general, the notion of constraint implies a model with an extended database. A constraint is a relation that limits the behavior of an entity or a group of entities. Examples of constraints are: a group of lines constrained to be parallel o perpendicular or collinear, a line constrained to be tangent to an arc, two cylinders constrained to be concentric, a dimension constrained to be less than a particular magnitude or equal to a multiple of a particular magnitude. The notion of constraint implies the notions of degree of freedom (DOF), overconstrained, and underconstrained models, as well as the notion of tolerance. A model can be conceptualized as a topological description of a complex form with n variables or independent dimensions. Each constraint diminishes the alternatives by one step. On the other hand, the bigger the number of constraints, the more difficult it is to manage in such a way that it will remain consistent under different values assigned to the remaining free dimensions. If a model is underconstrained, it cannot be resolved because some additional parameter must still be specified. If a model is overconstrained, it cannot be resolved because there is a contradiction somewhere. Constraint modeling requires that all the defined constraints shall be fulfilled before the model is evaluated or, in other words, that the DOF of the model have to be reduced to zero. The power of a system to deal with underconstrained or overconstrained models is a proof of its efficiency. Some programs inform the user that the model cannot be resolved but leave the user with the task of locating the fault. A program properly designed should have a constraint management module able to provide default parameters in case of an underconstrained model and to inform the user of this or any other contradictory parameter that may have been specified. Constraints can also be of two different types that sometimes are referred to as geometric constraints and physical or engineering constraints. Parallelism, perpendicularity, tangency, dimensionality are geometric constraints. But a model can also be based on formula like area = force/pressure. Constraints can also be specified as conditional relations such as: If $D1 + D2 > D3$ then $D1 = 10$ cm else $D1 = 20$ cm. A major difference between systems is the way in which the constraints are input and controlled. In general, this imposes some extra job on the user who, besides choosing an entity, marking its position and assigning some dimensions to it, must specify the relation that it shall keep with other entities in the model.

4. Evolution of parametric design techniques

Besides the above-mentioned pioneering work of Ivan Sutherland, Hillyard and Braid [1] proposed a system around 1978 that allowed the specification of geometric constraints between part co-ordinates in such a way that possible variations remain restricted to a range given by some particular tolerances. This proposal was not developed in the sense that could be expected from our present point-of-view. Gossard and Light [2] mention this work as a basis for their own, which can be quoted as the primary reference for what can be called parametric design in a more mature sense. The work of Gossard and Light that will commented below as a basis for what is called variational geometry or variational design, was a major step as it provided geometrical representations with new mathematical and geometrical tools that opened the way to the generalization of a model.

Around the end of the 1980s, when the main techniques of geometrical modeling, free-form surfaces and solid modeling were already assimilated, there was a growing sense that modeling techniques should advance in the direction of an increasing interactivity and ability to modify a model after it had been sketched. There were a number of important articles and books already published and, also, a few articles by researchers directly involved in the development of this field that attempted to resume the state-of-the-art. It is clear that there are still, at the present time, two big groups, one that is becoming obsolete and the other that attracts a growing number of researchers:

1. What we can call, as Roller [7] does, variants programming or static generation of alternative models by means of a programming procedure. These systems can rely on current internal representations of models.
2. Graphic generation or interactive methods by means of more elaborated systems that allow the modification of dimension and constraints after the model has been created. These systems imply a modification or an extension of the internal representation of the model.

The main disadvantage of the first group is that it cannot do what the second group does, that is, to change some of the characteristics of a model in an interactive way. On the other hand, it is a mode of work that can adapt to current CAD programs if the user has some knowledge of simple programming techniques. The main disadvantage of the second group is that we will have to wait a few years until a consistent parametric modeler, based on some of the different alternatives still under research enumerated below, is integrated in some of the programs currently used by architects.

4.1. Variants programming by macros or procedural modeling

One of the simplest ways of using a very rudimentary form of parametric design is to record a script of the commands and data values used to create an element. If this script is edited and the data values are changed, we will get a family of variants of the same type with different dimensions. We can refine this method by using a programming language, like AutoLisp, to write a macro, a routine or a little program that performs the suitable actions to model the element (the difference between these three terms can be assimilated to a difference in quantity, i.e., a few lines for a macro or a few pages for what might be called a simple program). In this way, the model can incorporate some kind of interaction with the user, that is, it can record the main parameters of the element as variables and request their values from the user once the program is activated. It can also incorporate conditional expressions or simple equations that may extend the interest of the method.

Variants programming is equivalent to one of the primitive forms of geometric modeling: primitive instancing. This consists also in the generation of models or elements by means of a procedure that call in sequence the commands needed to build the model. To prevent errors and secure the validity of the representation, values used for input have to be part of a pre-defined range. The main difference between this method and the ones that we will see below is that the commands used are already part of a CAD-modeler. The program reads the values as input from the user and executes the sequence of commands that create the model, which are provided by the modeler. The main limitations of this method are: the number and the range of variables is limited as there is, in general, no proper way for controlling variants that might produce not valid results. Moreover, the results cannot be edited; the only way to change the model is to repeat the process. However, it is a method of modeling widely used in industry and that can be very effective if one needs to incorporate to a model simple elements that are not supposed to be modified. It has been used extensively in architecture.

4.2. History-based constraint modelers

A graphically interactive parametric modeler allows the user to create a master model that can be used as a base to input parameters to the system and to request from the user the specification of constraints that will fix the model through a closed description of its components. This ensures that no errors should appear whenever any new variant is specified. As we have said before, this means that an extended or alternative internal representation must be used. There are different methods used to generate a new model after parameters are changed. The most widely used presently is perhaps what is sometimes called ''history-based design'' or proper ''parametric design'' (as opposed to variational design) or ''constructive parametric design''.

Many commercial parametric modelers available at the present time use a data structure keeping track of the sequence followed to create a model. Any operation, together with the data used to complete it, is recorded in the order that it occupied during the process of building a particular model. The operational parameters can be geometric entities as well as expressions. The model can be modified by substituting the data used in a particular operation. Recomputing the model will have the effect of changing some of its geometric characteristics while maintaining the connections, that is, the intended relations between the different entities. This method is also called constructive parametric design, as the sequence incorporates and requests directly from the user, the specification of secondary entities such as line or plane axes or a circle used to define an arc, etc. These constructive elements are also constrained in a similar way to the rest of the entities that constitute the model. One drawback of some commercial systems is that they try to present the constructive planes or axes as something offered to the user when it is rather something required by the system. Once the model is completed and DOF are fix to zero, and the model is neither underconstrained nor overconstrained, a construction plan is produced. The history is recorded by means of a directed graph where nodes represent entities and arcs operations. The direction of the graph follows the sense of constraint propagation. The result is a cyclic graph. To change a dimension is equivalent to substitute the values of the related geometric constraint. Adding a geometric relation is more complicated and requires checking possible overconstraints, finding the appropriate dimensional value and rebuilding the graph. The entities and operations involved, in general, must be such that can be constructed by rule and compass. Also, a single change in the procedure can force the system to recalculate, as average, a 50% of the geometry. In any case, once the graph is automatically reconstructed, the parameters are re-evaluated and the model is recomputed.

4.3. Variational geometry and variational design

Contrary to the previous method, parametric design based on variational geometry can recompute a design taking into account the actual situation, independently of the sequence that has been followed to reach this situation. The method relies on the description of parameters by means of equations and the availability of a system able to solve them. The main foundational reference is given by the articles published by Gossard and Light [2,3]. The method requires, as before, that the system is neither underconstrained nor overconstrained. The system has the advantage, contrary to the previous one, that it is independent of the way the model has been created and can accept any situation or any model as input. Dimensions are considered as constraints that affect a particular set of points in the model. An object in a space, defined by the three co-ordinates of their N-vertices will have $3N$ DOF. To compute the new geometry, after any of these vertices has changed, $3N$ equations (with $3N$ variables) will have to be solved. Many of these equations can be derived easily, such as the equation of a plane given by three vertices or any equation that forces four points to be coplanar. Similarly, simple quadrics like cones or spheres can be represented in this way. As distances are also implied, the constraint equations will be quadratic and are, for this reason, in general, non-linear. Numerical methods are used to solve them and the method proposed by Gossard and Light reduces the number of necessary equations to solve the model using a Jacobian matrix to find the parameters and a propagation systems that runs in both directions. Still, the method is computational expensive. It is doubtful, for the time being, that it could be applied to architectural models.

4.4. Rule-based variants: geometric reasoning by expert systems

The previous method has a number of important difficulties, such as the need to specify an exact number of constraints or to solve a large number of equations by numerical methods. To avoid these and other problems, a few alternatives have been proposed. Among them, some derived from Artificial Intelligence and Expert Systems can be quoted. Brüderlin [4], Sunde and Kallevik [5], Aldefeld [6] or Veroust et al. [9] are among the first and main researchers in this field.

These alternative techniques consider the model as something that can be described by a series of facts relating to geometric entities and constraints between them. A form is described by a series of logical predicates using languages like Prolog or

Lisp. A set of rules is specified to relate these constraints. These rules are applied through an inference engine that gives as output a particular model that satisfies the initial constraints and the production rules. The predicates can specify dimensions such as the distance between two lines (i.e., distance (l1, l2, d)) or geometrical relations (i.e., parallel (l1, l2) or on-line (p,l1)). The inference rules may look like this expression:

$$\text{on\_line}(p,l1), \text{on\_line}(p,l2), \text{not\_equal}(l1,l2)$$
$$\rightarrow p \text{ is intersect\_lines}(l1,l2)$$

deducing that point p is on the intersection of lines l1, l2, from the three predicates on the left of the expression.

The method is still under research. Direct specification of predicates is tedious, non-intuitive and error prone. It is also difficult to determine the number of constraints a particular form needs to be complete (‘‘uniqueness problem’’). There are some systems that help to solve these difficulties by means of additional predicates and geometric master models proposed by the user from which the set of initial implicit logical predicates can be automatically generated. Still the method is very expensive computationally and has to be refined.

4.5. Parametric feature-based design

What is a feature in CAD/CAM? A direct answer could be that a feature is something that can be extracted from a prismatic piece of material through a particular sequence of machine operations. Features are ‘‘slots’’, ‘‘holes’’, ‘‘blind holes’’ or ‘‘pockets’’, ‘‘chamfers’’, ‘‘fillets’’, ‘‘protrusions’’ and the like. But this is a CAM point-of-view that does not necessarily coincide with a CAD point-of-view, mainly during the first steps in the design process. From a more general standpoint, we can say that a feature is an entity that belongs to a semantic order higher than the geometric one. In literature, also the definition of feature as ‘‘a form with a defined function in a specific context’’ is found.

The first work in this field is probably a PhD by A.R. Grayer from Cambridge University, in 1976 (‘‘A Computer Link between Design and Manufacture’’) attempting to automate the relationship between a CAD- and a NC-system. This work comes from a group of researchers that had already worked in well-known modeling systems such as BUILD, ROMULUS or ACIS. By the middle of the 1980s, this topic had evolved until it became one of the most active fields of research in CAD/CAM. At the end of the 1980s, the first commercial prototypes supporting features and parametric design were available.

Features, in a parametric modeler, belong to families and are inserted in the model as instances of a master feature that is included in a features library. These features can be type-oriented or object-oriented. In the first case, the representation is given by attributes such as the geometric properties of the master feature (length, width, radius), tolerances, relations with other characteristics, etc. In the second case, the representation is based on procedures that process the main properties of the features. As the feature is defined externally and is inserted in the model during the process of design, the position of the feature must also be captured by parameters. Also, some features have natural counterfeatures, as what happens in CAD with, e.g., a gear rim that must fit into an inner gearing or as it could happen in architecture with a metal window that must fit into a hole in the wall. This means that the system must be able to support and manage the complex combination of relations of the model.

5. Architectural design and building models

The previous review was based on the available literature on Parametric Design and CAD. The kind of application that this literature is addressing has more to do with Mechanical Engineering and Computer-Aided Production, although there are a few papers related directly with the construction industry. But we are interested in architectural education and architectural practice and there are strong differences between these fields. The main one is that CAD, as used by engineers, applies mainly to objects that will be repeated many times and that are not rooted to any particular site. This is exactly the opposite of what happens in architecture. Keeping these differences in mind, let us now see how the previous review can relate to our field of interest.

(a) Variants programming is a well-established topic among the architects who use CAD and have some programming knowledge. Small elements, in 2D and 3D, can be created by means of some recorded procedure that requests the value of a few parameters when it is called by the user, activates the adequate commands from the modeler and produces and inserts a variant of a generic element on the master model. During the last few years, we have produced a wide collection of elements of this kind using AutoLisp. From such utilities as routines that open a hole in a wall, in 2D or 3D, and insert a pre-defined door or window, or routines that compute the dimensions and create a stair that runs from a given position on a floor to another position on the next floor, or routines that generate automatically other common building elements, etc., to other more speculative types that can, e.g., create geometrical compositions in a fixed or random way. We take it for granted that this kind of work is well-known, so we will not show any results of this kind in this paper. Some of these results will be published at our university and can be acquired by those interested. There is also a book by William Mitchell (*The Art of Computer Graphics Programming. A structured introduction for Architects and Designers*, 1987), that can still provide a fine introduction to this subject with routines written in Pascal but easily translated to other programming languages.

(b) Interactive parametric design of 3D elements can be achieved by means of a dialogue box with edit boxes that will allow the user to modify the values of the current parameters and include a graphical window pre-visualizing the results. Some commercial programs like ArchiCad or 3dStudio Max provide something similar to this but we are trying to implement it in a more general way through Visual C + + and programming tools like Arx. Anyone working in this field is invited to share experiences. In any case, we think that the kind of results is easily anticipated and we can accept that either by means of personal programs or by commercial facilities, there will soon be tools that will allow us to design small elements by means of parameters in an interactive way and insert them in an architectural model.

(c) The big challenge is the architectural model as a whole. There are two current positions concerning this topic. Some people think that it is not worth the trouble. Others think that any building can be parameterized as an assembly in a similar way as an industrial piece or a car or an aircraft. We are in an intermediate postion. It is doubtful that a building can be treated as an aircraft as this goes against the very nature of architecture that is rooted to a particular site and, besides, cannot repeat itself in order to justify fees and make our environment a little more interesting. Although it must be said that we have a lot to learn from the way these industries manage features and databases and parametric models. There is some research on specific methods to parameterize a whole building. A recent article by K. Martini presents an interesting program, still under development, based on an object-oriented class hierarchy, that can model a building through special primitives and positioners, using what is called intrinsic vs. contextual geometry in such a way that modifications of the model can be propagated to any part of it after it has been created.

It is clear, however, that any development of this kind will take a long time before architects can use it. Meanwhile, we are working on the following lines that we would like to present for discussion to anyone interested in the subject. An architectural model includes many different elements. We can start by considering only two main groups:

(a) Floors, which have holes, which have interior stairs, which have handrails.

(b) Walls, which have holes, which have windows or doors.

This relation is clearly hierarchical. Every handrail belongs to a stair. Every interior stair belongs to a hole in a floor. Every hole in a floor belongs to a floor. Similarly, every window or door belongs to a hole that belongs to a wall. Modeling of these elements should therefore be kept together on an object-oriented basis. If, e.g., a window is modeled as a feature that will be extracted from a public or private library and is attached to a hole, which is represented as a node in a tree graph and linked to a wall node, then any modification on any of the terms will be propagated to the others and we will not have to care about adjusting it. This can be done without much difficulty for small groups of elements. Things start to get more complicated when we consider a building with various floors and walls. Proposals as the one by Martini that we have commented above seem to be impractical for the time being. It seems to us leading beyond what is needed from a more practical point-of-view and from an immediate point-of-view and that, in any case, it will take too long a time to be completed.

A more promising approach might be considering a hierarchical subdivision of local co-ordinate systems and the fact that floors are horizontal and walls are, usually, vertical (we will not consider Gehry's buildings for a few years). Floors can be identified by a root (their local co-ordinate system origin). *Floor z-co-ordinates* can be kept in a separate table. Any modification on the floor height will automatically be propagated to every element linked to this floor through this table. Walls have a *z*-co-ordinate and a height that depend on the floors above and below. Any modification on the overall structure can be propagated through this table to all the floors that stand as global objects containing elements and sub-elements. The propagation of these changes to the whole structure by means of this general table cannot be avoided and may take some time. But the hierarchical organization outlined above will prevent propagation the other way round. This means that the elements in a stable part of the building can be modified more easily. The way these elements can be modified will be facilitated by means of some convenient interface as the one described in the previous (b) paragraph.

References

[1] R. Hillyard, I. Braid, Analysis of dimensions and tolerances in computer-aided mechanical design, Computer Aided Design 10 (3) (1978) 161–166.
[2] R. Light, D. Gossard, Variational Geometry in CAD, Computer Graphics, 15 (3) (1981), 1712–177.
[3] R. Light, D. Gossard, Modification of geometric models through variational geometry, Computer Aided Design 14 (1982) 4.
[4] B. Brüderlin, Using Prolog for constructing geometric objects defined by constraints, Proceedings of European Conference on Computer Algebra, 1985.
[5] G. Sunde, V. Kallevik, A dimension driven CAD system utilizing AI techniques in CAD. Report no. 860216-1, Senter for Industriforskning, 1987.
[6] B. Aldefeld, Variation of geometries based on a geometric-reasoning method, Computer Aided Design 20 (3) (1988) 117–126.
[7] D. Roller, A system for interactive variation design, in: J. Wozny (Ed.), Geometric Modeling for Product Engineering, Elsevier, 1989, pp. 207–219.
[8] C.M. Hoffmann, R. Juan, E-Rep. An editable high level representation for geometric design and analysis, in: Technical Report CSD-TR-92-055-CAPO Report CER-92-24, Department of Computer Science, Purdue University, 1992.
[9] A. Veroust, F. Schonek, D. Roller, Rule oriented method for parametrized computer-aided designs, Computer Aided Design 24 (10) (1992) 531–540.