A Perspective on Computer Aided Design after Four Decades Перспектива автоматизированного проектирования спустя четыре десятилетия

Earl Mark (Е. Марк),
The University of Virginia, USA
http://faculty.virginia.edu/mark/

Mark Gross (М. Гросс),
Carnegie Mellon University, USA
http://www.andrew.cmu.edu/user/mdg2,

Gabriela Goldschmidt (Г. Голдшмидт)
Technion, Israel
http://architecture.technion.ac.il

Introduction

It is said that to a man with a hammer the whole world is a nail. Our tools shape the way we see the world, and they shape the world we make. Computer aided architectural design tools are no different. They shape the way we see and reconstruct the built and natural environment. Studies have compared architects using traditional paper-based media with those using computer aided design applications, to see how the change in tools affects the designs. Some have focused on specific media, for example comparing desktop virtual reality with head mounted displays; 3D computer graphics with tangible interaction and so on. Here we are interested, not in comparing design media, but in looking at how software represents designs, and considering how the choice of representation can affect the kinds of designs and design processes that the software affords. We observe complementary approaches to design and propose that an open and loosely constructed technology environment will continue to build upon existing software technology and provide a fertile ground for further exploration and development.

The computer aided design tools that are used in professional design practice are the result of forty years of research, development, and commercialization. They have advanced from managing 2D line drawings displayed on vector screens to quite exquisite real time rendering of scenes with lighting, materials, and animation using raster scan technology. The interface has advanced, though less dramatically, from keyboard to mouse and windows to tablets and three-dimensional capture of physical models. The internal representations built into these tools have also advanced, from simply managing points, lines, and planes, to parametric and constraint representations such as those used by leading practitioners such as Foster and Partners, KPF, Morphosis, and Arup. To be sure, many of the ideas that are now widely adopted, or considered leading edge have a long history in research: building information modeling was conceived more than thirty years ago (Eastman 1987) likewise, capture of 3D input (Aish 1979, Frazer et al 1980); pens and tablets were in use from the earliest days (Mitchell 1977).

Until the fairly recent and still ongoing adoption of parametric and constraint based software in design, commercial computer aided design software supported designers in making and managing geometry. The computer was understood as a medium that the designer used to record and edit two- and three-dimensional form. Parametric and constraint based approaches add a layer in which the designer expresses relationships and dependencies among quantities (parameters) in the design. The commercialization of these technologies has begun to change architectural practice, and perhaps thereby also architectural thinking.

Researchers in computer-aided design are familiar with a range of approaches to computer-aided design that have been explored by the research community but not (yet) been developed and adopted commercially. We believe the lack of adoption in practice does not, in most cases, reflect the poverty of the approach, but is due to marketplace factors that may be on the threshold of change. Each approach, if and when it is adopted, might have a profound effect on architectural design thinking and practice as parametric design has had in recent years. Therefore in the rest of this paper we review briefly several of these approaches to design and reflect on the underlying models of design that they embody.

First Generation Tools

First generation computer aided design tools were intended to support a design process rather than provide for pre-conceived and ready-made solutions. Sketchpad, the 1963 landmark system developed by Ivan Sutherland at MIT (Fig. 1), demonstrated that a design process could be viewed as setting and relaxing geometrical constraints, building up a kit of object instances, and allowing for parametric control over their formation (Sutherland 1963).

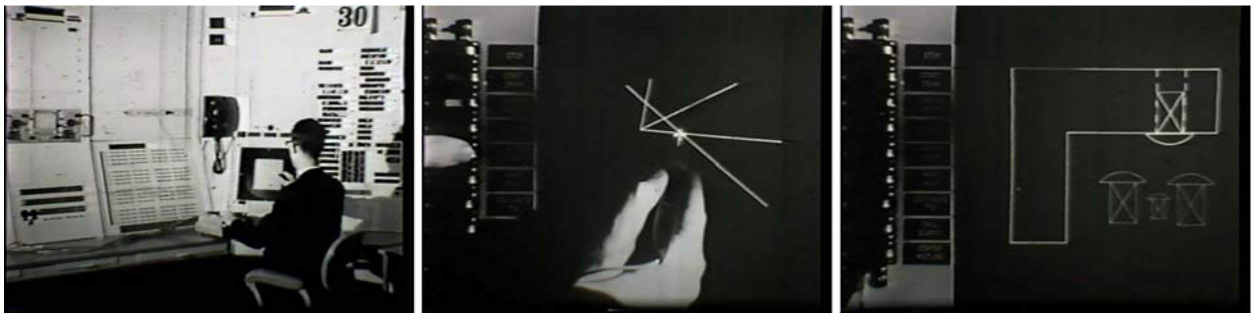

Fig. 1. Ivan Sutherland's Sketchpad system

Some later generation tools, in order to help automate design and documentation, focus constraints, instances and their variation on conventional architectural types, such as "wall types", "window types" and other predefined objects. These systems appear biased towards commonly recognized building products. Still, their intention is to give the designer a free hand in the layout, scale, and relationships between a set of parts.

On the one hand, designers value flexibility in exploring form. On the other hand, they also constrain objects as they transform a sketch into a representation invested with specifically constructed materials and purposes. The state of the art is imperfect in addressing these at times contradictory demands. For example, some tools focus on geometry only and allow for a great range of formal explorations. Other more automated computer aided design systems offer design lifecycle features that can quickly expedite assemblies of pre-defined architectural objects based upon limited libraries of architectural components (e.g., doors, windows, walls). We can join two walls together automatically provided that the software anticipates geometrical configurations, material attributes, and part-to-part relationships.

Today it is difficult to find application software that provides both a capacity for open geometrical modeling and automated assembly of standard architectural objects at the push of a button (e.g., Pella Windows, etc.). The automation works if the software designers anticipate the geometry of the objects that they are required to accommodate. Can we "train" a computer aided design system, as first proposed by Negroponte, to become more adept at enabling explicit architectural forms while still allowing for a more open geometrical description of objects, and, going further, to partner with us in defining a design process that is customized to each new design problem (Negroponte 1970)? The range of approaches and opinions varies widely.

Eight approaches to design

We consider eight approaches to design through the lens of different ways of constructing computer-aided design software. We have mentioned the first - parametric and constraint based design - which (as a basis of both building information modeling and digital fabrication) is already transforming architectural design practice. The others are: shape grammars, frame based design methods, object oriented design, generative systems, top-down design, knowledge based design systems, and design and cognition. These categories overlap and several may draw upon the same methods. However, we use them to draw out some distinct approaches. Although most readers will be familiar with these approaches we offer a summary of each.

Design by Constraints

One can view architectural design as making a building to satisfy constraints and achieve objectives. Among the constraints are site, construction, and program requirements such as "it must be on this site", "it must stand up", and "it must provide working, play, and living space for one hundred people". Although the word "constraint" suggests something restrictive, a constraint can be anything the architect wants the building to satisfy. An objective is like a constraint, but to be achieved to the greatest extent possible. "As sunny a living room as possible", "The smallest possible building footprint", and so on.

Engineers think about design this way: as a problem to optimize an objective function while satisfying constraints. State the constraints and the objective function and you can apply mathematical machinery to crunch numbers and produce the solution. For example, designing an airplane wing entails producing the shape that gives maximum lift for minimum weight, subject to constraints on strength of materials. In architecture, we seldom can state the constraints and objectives at the outset. We develop them as we go: problem and solution co-evolve. And the constraints and objectives of architectural design aren't so easily stated as "maximum lift".

But if we can describe some constraints - sizes of things, spatial relationships among built and open elements in the building - then we can program a computer to manage them. Constraint-based design software records not only sizes and positions of physical elements but also the relationships that we want in the design. This window is in this wall. These two rooms are adjacent. When we change the design - move this column, resize a room – the software keeps the relationship we stated. Design by constraints views design as managing a large and complicated set of relationships.

A simpler variation is parametric design in which some quantities (dimensions, positions and numbers of elements) are expressed as a mathematical function of others. For example, the number of windows depends on the length of the wall; the longer the wall, the more windows. Parametric design sets up these dependent relationships, and when the designer changes key driving variables, dependent ones follow. It's like design by constraints, except that the dependencies only go one way. Parametric design was championed in the 1980s and today many CAAD application offer the ability to relate one quantity to another.

Shape Grammar

Shape Grammar (Stiny 1980) derives from the work of mathematician Emil Post (Davis 1989), whose production systems form the basis of modern computer language parsers as well as rule-based expert systems in artificial intelligence. Shape Grammar trades on the idea of a "language of form", and the use of grammar to describe the structure of natural language. Languages have syntax and semantics. If we speak of languages of form, a shape grammar describes its syntax.

The key idea in shape grammar is the shape replacement rule that says, "whenever you see this shape, you may replace it by that shape". A shape grammar is a set of replacement rules. Grammars can be designed to begin with a single dot, to be replaced by a line, the line by a rectangle, and so on until quite complicated forms appear. A carefully designed grammar can capture a design style, as many examples show. Researchers have designed grammars that produce all manner of forms: Japanese traditional tea houses, Frank Lloyd Wright prairie houses, coffee makers, soda bottles, microelectronic mechanical systems, and plant forms. By exercising production rules, one after another in different sequences, a grammar generates a different variant within a universe of forms.

Frame based design methods

The key idea in frame-based design is that design proceeds by retrieving and specifying design "frames" (Minsky 1986), each of which consists of a collection of "slots" that represent attributes. The frame (like a prototype, object instance, or even a case) represents knowledge about a typical design. For example, the frame for "school" might include slots for classrooms, playground, principal's office, teacher's lounge, and auditorium. When the design software "instantiates" the frame, it brings into play previous knowledge about schools: what to expect in a school design and what decisions must be made. Some slots may hold default values, for example: unless the designer specifies otherwise, a classroom will hold approximately 30 children. A frame based design system has three main parts: a mechanism to retrieve frames from memory that match – more or less - the design problem at hand; a means to adapt the frame by adding detail or overriding defaults, specifying the frame for the problem - and a mechanism to store new designs as frames. A fully realized frame system would enable the designer to retrieve, apply, and combine frames at different levels and aspects of design. For example, the school design might be composed of a frame for site design, a frame for structural design, frames for classrooms, offices, and play areas, and so on. Case based design (Oxman 1993; Zimring et al 1995; Heylighen and Neuckermans 2001) can be considered a kind of frame-based design. The idea is that a designer seldom works from scratch. Rather, a designer begins with a known example (a case, or precedent) of the desired design, and then adapts the case to respond to the particularities of the problem at hand. For example, an architect designing a school might begin with a known school design, and then make changes to the design based on site, program, and other local conditions.

Object oriented design

The paradigm of object oriented design derives from the basic concept that an object carries within it the processes it requires to adapt or situate itself in response to information that is passed along to it.

Object oriented code thus consists of objects that contain knowledge about the processes and data needed to define it within a larger programming language. Correspondingly, an object oriented computer aided design system might allow a designer to think in terms of the placement of objects that are adaptive rather than need to be fully instructed as to how they react. The approach includes classes of objects that perform according to inherited properties and behaviors. Within more complex and some commercially successful systems (e.g., Revit), object oriented design includes objects that contain information about cost, specification of materials, and geometry that is adaptive to design process and documentation strategies used in practice.

While object oriented approaches have reached a level of industry wide use and acceptance, implementations vary widely. Some systems afford an open ended environment in which the definition of architecture objects is relatively fluid and unencumbered by pre-defined rules. Other systems capture industry knowledge and processes at the cost of limits to their flexibility. These later systems require a large investment in structuring formal and spatial relationships that may not fit the quickly changing perspectives that occur in schematic design. The ability to dynamically create objects within such a system then becomes a limiting factor on use.

Generative Systems

In generative systems, a set of rules or formulae establishes the framework for a computer based production process. The author-designer sets up the rules and the resulting computer process generates shapes and designs without the designer individually determining each solution. For example, Bentley Systems' relatively new commercial system, *Generative Components* sets into play a series of user defined transactions and parameters, generates forms, and then allows the designer to make dynamic modifications in order to adjust the results. Most generative systems focus on form generation or solving specific structural or space planning problems, but some also include setting up design evaluation or feedback systems. Within generative systems, strategies include those found in shape grammars, such as rule-based systems, and also strategies associated with design and cognition, such as neural network approaches. Researchers have also applied generative systems retroactively to reviewing a past project (Caldas and Rocha 2001). A celebrated example is Foster and Partners' Swiss Re Tower in London, where so-called "dimension driven cells" were rigged and then parameters varied to realize the distinct flat diamond geometry of the exterior of the skin.

Top Down Design

In top down design an abstract overview is the beginning point towards realizing a set of objectives. In architecture, a top-down methodology might consist of a larger formal massing study before any specific materials or more detailed development of objects begins. In product design an overview assembly of a project is developed at the beginning of a design process prior to sub-assembly and thinking through details on a finer grained level. The Pro/Engineer software is a commercially used technology that has been optimized for this approach.

Yet, in practice, the top down approach is not delivered in absolute terms. The technology may provide for libraries of parts, parametric approaches, and additional techniques associated with other methods. In "The Art of Computer Graphics Programming: A Structured Introduction for Architects and Designers", Mitchell, Liggett, and Kvan (1987) developed a top-down adaptation of Turbo Pascal for architectural case studies. The structuring of the case studies is based upon top-down reasoning and methods of exercising parameters that control the size, number and placement of elements. For example, a classical column is initiated with a blank template, and then, as a layout is developed, is further detailed as Doric, Ionic or an alternative type. The later part of the production process thus considers the detailed development of individual components. Working with such a system, the designer can still engage in a combination of top-down and bottom-up thinking. Yet, the prevalent direction of the exercises places value on top-down considerations in assembly. If such an approach is applied too rigidly, the discovery of problems at the detailed level that require reconsideration of the entire top-down structure may occur later than it would in a more balanced approach. Conversely a designer who becomes lost in the details, or takes on details at too early a stage, may fail to see larger schematic options. Optimized use of this method is then a process of reconciling larger assemblies with detailed elements in a mutually adjustable process.

Knowledge Based Design Systems

Knowledge based design systems (Coyne et al 1989) have the implicit premise that a design process does not begin *tabula rasa*, but rather builds upon what has come before, either in precedents for the project or in the history of a design project itself. Capturing and flagging this information appropriately may remind the designer of considerations related to current design decisions. In this approach, specially trained knowledge engineers elicit knowledge from domain experts to develop the software. In some implementations, the system uses the acquired knowledge base to generate candidate formal or spatial layouts for the designer to select from.

Still, design activity may require a response to a distinct set of problems that calls upon relatively novel approaches and solutions. Building a knowledge based design system in such circumstances is more challenging. Varied techniques have included rule-based if-then statements for capturing knowledge, neural-nets where nodes and connections are made to capture knowledge, and fuzzy logic for handling imprecise circumstances. Still, individual variation in design method is complex and is probably difficult for even the best system to generalize. A number of implementations are devoted to solving specific types of design problems, e.g., a knowledge based system to assist in space-frame design.

Design and cognition

The study of design and cognition bridges the disciplines of cognitive science, artificial intelligence and computer based design methods. Research methods include capturing and observing human design activity. It includes methods of form finding and reasoning. Scholarship can include computer generated agent based activities, providing insights into human interaction with design tools or architectural places. Advances in collaborative design methods are also cited in this area. Thus, design and cognition often draws upon, feeds into, or overlaps with technologies associated with the first seven areas. The work varies widely from speculating upon or evaluating human behavior to the incorporation of observable design problem solving paradigms into computer aided design tools.

Observations

Reflecting upon these approaches to computer aided architectural design, we find common ground in some observations:

1. Design process in architecture is intimately associated with geometric modeling.

2. Geometric models must be mutable with less predefined representations.

3. Variation does not proceed only at the whimsy of the designer but is informed and can be controlled via a set of established relations between objects and invented rules

4. A design proceeds from ambiguous and loosely defined relationships among objects to deliberately associated relationships.

This yields some cautiously offered propositions about the next generation of tools:

1. Geometric modeling processes should at times re-engage the histories of how individual objects have been created.

2. Knowledge-based approaches or rule-based systems must not be fully deterministic.

3. The development of smarter design tools will continue to beckon research and theory well into the future, as each generation discovers new paradigms to more tightly connect design activity with the acts of making and building.

Given our present view of the state of the art in design tools, what direction might development of technology take? In this brief survey we have attempted to summarize a pattern of achievements over the divergent history of computer-aided design. We propose that the next generation of design tools emphasize diversity of specialized approaches rather than comprehensive technology solutions. Design and Cognition is not so much one of a distinct number of methodologies, but perhaps guides a great combination of working methods and strategies.

Martin Woolley (2004) argued that in the “post- IT era” tools must be developed ad hoc as needed, where (unlike today) the designer is the tool builder. In this respect the tool is not selected, but created contingent on the type of design task, the stage it is in, and adapted to circumstances. This may have more implications for interfaces than for the actual output of the particular tool, for databases that are fluid, and for tools that offer an open architecture, rather than those that make automated construction simpler and easier for all.

The growing acceptance of computer technology in the discipline and profession of architecture has motivated putting the present state of the art in perspective. The most significant contribution of the first generation of computer aided design tools was to set into place a kind of geometry embodied with reason. The first such system recognized the value of constraints, parametric variation and instances in the deliberative processes required of a designer. The history of the technology has been enriched by varied attempts to articulate the reasoning used to construct a geometric model. In this sense, computer aided design allows for the highest-level discussion of intention. We predict it will continue to develop according to the ways in which we are able to expand upon the means to construct geometry. This may occur through increasing integration of computer aided geometrical modeling and direct physical form making (e.g., CNC fabrication). Or, it may occur through a growing number of techniques (e.g., scripting, associative geometrical modeling tools and constraint managers) that empower the end user with an increased ability to independently construct a model of the design.

Acknowledgement

We thank Alexander Koutamanis for providing the incentive to write this article as part of a planned book on “the architectural image and the computer.” Due to the tragic fire that consumed потребляемый Alexander’s office in the Bouwkunde at the TU Delft, the book project is no longer active.

References

Aish, R.: 1979, 3D input for CAAD systems, Computer-Aided Design.

Caldas and Rocha: 2001, A Generative Design System Applied to Siza’s School Of Architecture at Oporto, Proceedings of CAADRIA, 2001, pp. 253-264.

Coyne, R. D., Rosenman, M. A., Radford, A. D., Balachandran, M. and Gero, J. S. (eds.): 1989, Knowledge-Based Design Systems, Addison-Wesley Longman.

Davis, M.: 1989, Proceedings., Fourth Annual Symposium on Logic in Computer Science, LICS ‘89, IEEE.

Eastman, C., Lividini, J. and Stoker, D.: 1975, A Database for Designing Large Physical Systems, in Proceedings of the 1975 National Computer Conference, AFIPS Press, Montvale NJ.

Eastman, C.: 1986, Fundamental Problems in the Development of Computer Based Architectural Models, Proceedings of the Computers & Design Research Symposium, M.I.T., August 4.

Frazer, J. H., Frazer, J. M. and Frazer, P. A.: 1980, Intelligent Physical Three-Dimensional Modelling System, Computer Graphics 80 Conference, Conference Proceedings, Online Publications, pp. 359-70.

Heylighen, A. and Neuckermans, H.: 2001, A case base of Case-Based Design tools for architecture. Computer-Aided Design, 33(14), pp. 1111-1122.

Minsky: 1986, Society of Mind, MIT Press.

Mitchell, W. J.: 1977, Computer-Aided Architectural Design. Petrocelli/Charter, New York.

Mitchell, Liggett and Kvan: 1987, The Art of Computer Graphics Programming: A Structured Introduction for Architects and Designers, Van Nostrand Reinhold.

Negroponte, N.: 1970, The Architecture Machine, MIT Press.

Negroponte, N.: 1975, Reflections on Computer Aids to Design and Architecture, Petrocelli/Charter.

Oxman, R.: 1993, PRECEDENTS: Memory structure in design case libraries, in CAAD Futures ‘93, Elsevier Science Publishers.

Schon, D.: 1983, The Reflective Practitioner: How Professionals Think in Action, Basic Books.

Stiny, G.: 1980, Introduction to Shape and Shape Grammars, Environment and Planning B, 7, pp. 343-351.

Sutherland, I.: 1963, SKETCHPAD: A Man-Machine Graphical Communication System, Proceedings of the AFIPS Spring Joint Computer Conference, Detroit, Michigan, May 21-23, pp. 329-346.

Woolley, M.: 2004, The thoughtful mark maker, in G. Goldschmidt and W. L. Porter (eds.), Design Representation, Springer Verlag, London, pp. 185-201.

Zimring, C. M., Do, E., Domeshek, E. and Kolodner, J.: 1995, Supporting case-study use in design education: A computational case-based design aid for architecture. in J. P. Mohesen (ed.), Computing in Civil Engineering, American Society of Civil Engineers, New York, pp. 1635-1642.