What has yet to be CAD

by Chris Yessios
Department of Architecture
The Ohio State University

The theme of this Acadia Conference was to a large extent addressed by Mitchell in his article "What was Computer-aided design?", published about two years ago [1]. While one has to agree with most of his points, I find his predictions gloomy enough to wish I could disagree. Luckily, Mitchell has chosen to address what the majority of the profession (and many architectural schools) currently consider to be CAD. It turns out that this CAD is not what CAD is supposed to be. I have, therefore, purposely chosen a title which appears to echo an opposite view. My intention is not to express disagreement but rather to project the other face of CAD, in my own mind, the only CAD which deserves the name.

Whether the current CAD should or will be called CAD in the future is of non-essential significance. As teachers of architectural design we need to be concerned that architectural CAD remains, to date, a very immature field. It is CAD only by name, since a true CAD system has yet to be "discovered".

This presentation consists of three major sections. The first reviews why the currently available CAD systems do not have the ingredients which may justify them as design oriented machines. This discussion leads to the identification of architectural modeling and knowledge systems as the two main areas which need to be researched so that they may offer the basis for the development of truly design oriented machines. Each is discussed under a separate section, but the point is also made that the two should work hand-in-hand and should be integrated into a completely unified system.

With respect to the theme of this Acadia Conference, this presentation, rather than attempting to predict what might happen in the next decade, it aims at identifying a few critically significant areas which need to be researched if CAD systems worth their name are to be produced. The list of areas covered is not exhaustive. Those discussed reflect work currently in progress at the Ohio State University and some relevant examples of ongoing work will also be shown.

In my concluding remarks I also take the liberty to encourage those architectural schools which are currently "mere passive consumers of commercial computer technology" [1] to get involved with researching what a true architectural design machine ought to be. Let us keep in mind Wiener's warning of some thirty years ago that the computer is a "double-edged sword" which can either benefit or damage us, depending on how we choose to use it [2].

Why architectural CAD is not yet CAD

In spite of the fact that I expect the majority of the audience to agree with this position and I could, therefore, take it as an axiomatic truth, it is worth reviewing shortly what CAD is supposed to be, to simply uncover the fact that it currently has none of the ingredients it was meant to have.

What is (or should be) CAD: By definition, "Computer Aided Design" (CAD) is assumed to be addressing "design" and it is assumed to be "aiding" it. A quick look at how the profession, as well as many architectural schools, practice CAD today suffices to verify that it is neither design oriented nor an enhancement to design. Unfortunately, the initial 'D' can also stand for and at the same time conveniently hide the term 'drafting', which is what currently available systems are, even when
they are called CADD, in an effort to include both design and drafting. In the few cases where attempts are made to use CAD in design, the result is frequently "computer hampered" design.

But some will raise the argument: why even make the distinction, since design includes drafting. I shall make no attempt to directly address this issue, neither shall I attempt to define the whole spectrum of design activities. After all design means different things to different people. I shall suggest a criterium to be used for distinguishing what I shall propose to call design and what drafting.

**How to distinguish design from drafting:** Design includes all activities which occur before a solution, final or preliminary, exists. It includes activities such as problem solving, decision making, value judging, conceptualization, information retrieval, and compositional creativity, where the list is not exhaustive. After a solution, preliminary or final, has been decided upon, it needs to be externalized and communicated either for visualization or construction purposes. The latter is done through drafting. Drafting is not involved with any problem solving oriented decision making, other than possibly deciding what line weights are to be used in what parts of a drawing.

Independently of what different people want to call design and what drafting, the fact remains that there exist those creative processes during which an architectural design solution, or parts of it, is still very tentative and we need the ability and the ease to further develop it, revise it, analyze it, question it, and in general manipulate it as a very soft and dynamic entity. When it becomes hard, then we are ready to draft it. I know of no CAD system which is capable of accommodating the dynamically tentative stages of architectural design.

**Design versus visualization:** Above, I have broadened the definition of drafting enough to include visualization. Since to many the ability to visualize design alternatives is a critical aspect of design, the issue needs further clarification. Even more so when we recall Greenberg's suggestion [3] that the current computer graphics research leading to the production of realistic images constitutes the "breakthrough" required for making a significant use of the computer in the practice of architecture.

We simply apply the criterium proposed earlier. If the visuals are direct byproducts of design oriented procedures, then they themselves are design procedures. If, instead, they require a special programming effort to generate the "data" which can be plotted as an image, they do not constitute design in a true sense, no matter how realistic that image might be.

From a different point of view, the time required for the generation of an image is of major significance. As design proceeds there is a need to visually inspect the current state of an artifact. But such visualization has to be virtually instantaneous. By current practices, visualization is performed after the solution is known. It takes a significant effort to produce the data and it may take a few hours to plot the image, whenever a certain degree of realism is sought. Without denying the value of visualization for the presentation of finished products, its value as a design tool which enhances creativity has to be questioned. Just as slow reading permits to the brain to get lazy, waiting for a visual to be produced disrupts the creative process of design. Consequently, unless visualization is a direct extension of the design procedures, utilizes the same internal structures which the design process operates upon and reflects the tentative, soft and dynamic character of design, then it does not satisfy the requirements for being labeled a design procedure.

**Design versus solid modeling:** Quite sometime ago most of us realized and agreed that, the product of architectural design being a three-dimensional physical entity, it can best be accommodated by geometric modeling procedures. We then fell in love with solid modeling and particularly its set theoretic operations of union, intersection, and difference.

In solid modeling, basic primitives such as cuboids, cylinders, and/or spheres are generated and then composed together either by simple arrangement operations or by applying some combination of the set theoretic operations. The corresponding manual method is when we construct physical scale models by cutting and pasting cardboard, wood, or other materials. When we keep this analogy in mind, the main drawback of solid modeling as an architectural design tool becomes apparent. Try to change a physical scale model. The only way to do it is by rebuilding the model itself. The situation is not much better when using computer implemented solid modeling methods.

Applying the design/drafting criterium proposed earlier, one has to observe that solid modeling, at least from the point of view of architectural design, is a drafting method. It can only be used effectively when the building to be modeled has already been designed. It is very cumbersome to use
during the tentative stages of the design process, since it cannot be responsive to the semantics of architectural design oriented operations.

If this is not CAD, what is? Having applied the design/drafting criterium, it did not take much to conclude that what today is promoted as CAD, including solid modeling, is not capable of accommodating those stages of design where a solution is still unknown and it is being searched. Assuming we have agreed that, to accept a CAD system as a design oriented machine, it needs to be able to facilitate architectural design during its tentative and soft stages, what then are the capabilities which we need to research and develop?

Firstly, let me outline two general conditions which can also be used as criteria for evaluating the directions of our research efforts:

a. Whatever a design oriented CAD system is or could be, it should be implementable today or in the very near future. Utopian systems such as the Architecture Machine [4] proposed about fifteen years ago, while useful as long term perspectives, raise more questions than they answer and tend to disorient our research priorities, when taken literally.

b. A design oriented CAD system will be worth its name when and only when it can be proven that it helped us produce better designs. That it enhanced our creative processes in ways such that we were able to discover design solutions and architectural configurations which we would be unable to do with conventional methods.

Next, I shall propose that a design oriented architectural CAD system should be based on a yet to be fully developed theory of architectural modeling. It should facilitate and allow building models to behave like buildings when they are operated upon. It should also facilitate the incorporation of previous architectural experiences and design theories without restricting innovative paths of discovery. These capabilities should be fully integrated.

Even though I purpose that architectural modeling and knowledge systems should be fully integrated, I shall, here, discuss them separately, following a current common practice. The issue of modeling has been around for a rather long time, but we have yet to address it in an architecturally meaningful manner. The proposition that previous knowledge plays a critical role in architectural problem solving was also first presented quite some time ago but it gained acceptability only recently. Architectural researchers still tend to see the two areas as, may be complimentary but quite different [5]. I find this attitude objectionable.

In the following sections, I discuss general principles rather than specific solutions, even though a few specific examples are mentioned. My main goal is to identify a few areas which are believed to be of critical importance towards our efforts to construct architectural design oriented machines.

Architectural Modeling

While the majority of the currently available commercial solid modeling systems have been imported from Europe, where their development was triggered by the pioneering work of I. C. Braid [6], in this country, architectural researchers were among the first to make significant contributions to modeling research (C-MU’s Eastman et al., [7] and Michigan’s Borkin et al., [8]). Yet, in some ironic manner, we failed to recognize the inefficiencies of solid modeling for architectural design. More than that we recently appear to claim that the work on modeling is “done” and the field is worth no more research efforts. I have to argue that the contrary is true. We have barely scratched the surface of modeling, at least the type of modeling which is capable of facilitating and enhancing architectural design.

Extending the general definition of modeling, architectural modeling should be a body of theory, methods, and operations which (a) facilitate the generation of informationally complete architectural models and (b) allows them to behave according to their distinct architectural properties and attributes when they are operated upon. I shall next elaborate further on the most important requirements implied by the above definition.

Facilitating the generation of architectural models: This requirement has two interrelated parts: The first is that we be given the means to generate architectural models through automatic procedures, which recognize architectural symbolisms and have sufficient intelligence to produce complete
models with minimum input by the user. For example, floor plan sketching procedures lending themselves to the automatic generation of 3-D building models.

The second part of the requirement is that our generative procedures ought to be capable of accommodating design during its tentative stages. The generation of an architectural model cannot be viewed as a single-pass sequential procedure, but as a process which is as flexible and soft as the artifact being designed ought to be, before a certain solution, preliminary or final, has been derived. For both parts of this requirement, current solid modeling practices offer us examples to be avoided.

Adapting CAD to familiar design procedures and architectural semantics: Any suggestion that architectural designers should be "retrained" to be able to take advantage of the powers of the computer has to be considered absurd. We, therefore, appear to have no choice but to model our modeling methods after methods known and understood in the practice of architecture. But there are more reasons for doing so. Those conventional methods have served architecture well for a number of centuries, their "wisdom" has been tested and have at times produced "great" architecture. We should certainly allow these methods to evolve as the new powers of the computer are assimilated, but at no time should we even talk about "revolutionizing" (meaning: radically changing overnight) the traditional methods of architectural design.

Modeling our computer implemented design procedures after methods familiar in practice should not restrict us from taking full advantage of the power offered by the computer to add some "magic". Nothing which the human designer cannot do, given enough time. Even simple operations such as accurately dimensioned double line wall representations with properly "cleaned" corners become magical when they happen instantaneously, something which the computer can do well. The really magical effect of such features relates, of course, to problem solving. Instantaneous visualization of design operations, whether in 2-D or 3-D, facilitate an uninterrupted flow of the mental processes of design.

Semantically consistent operational behavior: During generation time, as well as after an architectural model has been generated, the architectural elements it is made up from need to be able to behave according to their attributes, when they are operated upon. Note that solid modeling cannot accommodate such a requirement, since all its primitives are of the same type: solids. Externally attaching lists of properties to function as constraints, when a certain element is operated upon, would, in theory, be an improvement which would allow solid models to simulate the behavior of architectural models. But it would certainly be a very inefficient and computationally expensive method.

It is possible to develop and implement internal structures which can directly incorporate the applicable attributes, rather than requiring long lists of properties and rules of behavior. As an example I shall mention the void modeling structures we have developed at OSU, which have proven highly efficient in accommodating the operational behavior of architectural elements. Without question much more work is needed in this area.

Levels of abstraction: Any entity we model can be viewed at different levels of abstraction or detail. Actually, in any model it is desirable to be able to incorporate only as much detail as required by the targeted application or analysis. In a similar fashion architectural models need to be constructed at different levels of abstraction. The most commonly required levels are the following:

a. The volume level, where buildings and other elements are represented as solid volumes and with very little detail.

b. The building level, where each building is modelled with its walls and spaces and all the other details which completely describe its form and function.

c. The construction level, where each building element, such as a wall, is shown with all its structural details.

The architectural designer requires the capability to work at any of the above levels of abstraction, depending upon the current stage of the design process and his/her personal preferences. As design work proceeds at one level of abstraction, models at the other levels should be automatically generated and be readily available.

Actually, my wording above may be misleading since what is really required and proposed is a single internal structure from which models at different levels of abstraction may be extracted. Thus, this
requirement is no different from the rather widely recognized requirement for 2-D/3-D integration. 2-D representations should be considered yet another level of modeling abstraction.

**Automation versus Interactive problem solving:** The term "automation" is here used in its true sense and not as in the promotional material of commercial CAD systems, where everything appears to be automated, when in fact it is very manual. The real issue here is whether or not the machine ought to be entrusted with automatically providing solutions, or should this remain the realm of the human user and designer.

As a general principle, I propose that the machine be allowed to automatically execute as much as it can, but never accept its outcome as a "god" given truth. Always allow the user to have the final decision on whether or not what the machine did is acceptable. As it turns out, there are tasks which the machine can execute more efficiently than a human, where the human user still has the option to provide specific instructions and directions. For example, the extraction of models at different levels of detail can be done by the machine faster and more accurately. There are, therefore, no reasons why such tasks cannot and should not be assigned to the machine. As a matter of fact this is one of the major drawbacks of commercial CAD systems. In spite of their claims for automation, they simply do not incorporate enough.

On the other hand there are tasks where the machine is expected to do a poor job. These are tasks which involve problem solving and require value judgement. The failure of attempts to automate space planning is a well documented example. But even in these cases the machine can be allowed to do as much as it can and to function as an "idee" generator or brainstorming device. Whether this is problem solving in a true sense or simply the extraction of pre-stored knowledge and experience, possibly presented after the application of some generative schema, is an issue which goes beyond this discussion. Whatever the case, this is an area of capabilities which we have yet to sufficiently research.

**Specificity versus generality or how to accommodate individual styles:** Any degree of automation and any type of object specific internal structure implies a "bias" towards a certain style of design. I have already proposed as a general requirement that the user be allowed to further manipulate whatever is automatically generated by the machine. While this allows a certain degree of flexibility, it is always from within a specific framework from which design is approached. For example, the floor plan sketching procedures, which lend themselves to the automatic generation of building models, imply that a certain designer is willing to approach his/her design through floor plan sketching. While many do so, many do not.

I shall claim that the number of methods by which design can be approached is quite finite and that respective automatic procedures can and should be provided for each one of them. In another paper [9] an attempt to categorize them under two general types has been presented. But we can only hope to automate those procedures which are known to be in use in practice. This by necessity excludes the possibility of innovative discoveries which individual designers might have a need for. Thus a general method for approaching design is also required. To avoid biases, this can incorporate very little automation and will, therefore, be rather tedious to use. Once again, we have yet to sufficiently research both specific and general methods for generating architectural designs.

**Knowledge Systems**

Architectural Knowledge system ought to be an integral part of modeling procedures and of any design oriented CAD system. As I already noted, the fact that I have chosen to discuss them under a separate section, should not be taken to imply otherwise. Secondly, it needs to be pointed out that the need to incorporate knowledge in architectural design oriented procedures is not a new discovery as one might be led to believe, given the current popularity which has been triggered by the recent commercial success of expert systems, primarily in diagnostic applications. Even the diagnostic expert systems themselves are at least ten years old, but have entered the spectrum of architectural research interests rather recently. Once again, we appear to be fully embracing models of expert systems which were developed for different fields, rather than researching the roots of the architectural problem; what knowledge means in architectural design and how to acquire it.

**Problem solving and knowledge systems:** The field of Artificial Intelligence (AI) has studied problem solving extensively in its efforts to develop automatic systems which perform tasks which are con-

What has yet to be CAD
sidered intelligent when performed by humans. At first AI attempted to study and simulate the neurons of the brain. When this approach failed, it attempted to simulate the problem solving behavior of humans after studying it through the analysis of problem solving protocols. The main emphasis was on the development of heuristic search techniques, most of which were modeled after problem solving "tricks" observed in humans. While these efforts produced valuable theoretical results, from a practical point of view they produced systems which were hardly sufficient as problem solvers. Only when AI recognized the role of knowledge in human problem solving, it was led to the development of expert systems, which have proven highly efficient, primarily in diagnostic areas.

A typical expert system consists of a knowledge base, a knowledge acquisition module, an inference engine, and an explanatory facility. The inference engine processes symbolic information stored in the knowledge base and is capable of deriving conclusions at certain explicitly stated degrees of likelihood. A major feature of the expert systems is their ability to explain how their conclusions have been derived, which represents a major deviation from the previous "black box" tradition of AI. A major research issue with expert systems is the acquisition of knowledge and how to automate it.

The AI research on problem solving has had a major influence in architectural design research and offered the theoretical basis for the attempts to automate space planning and spatial design. But this work did not produce credible results since it ignored the role of previous experiences and knowledge. Alexander, being one of the first to introduce algorithmic problem solving through his decomposition-composition procedures [10], he also was the first to introduce the concept of previous knowledge through his work on pattern languages [11]. The theory behind the pattern languages is that architectural designers do not problem solve from scratch as if each architectural problem were a completely "new" problem and no previous experience exists. On the contrary, known "patterns" of previously applied solutions become the basis for problem solving. In an interesting manner, by the time the pattern languages were proposed, Alexander was rather disenchanted with computers and his pattern languages were never developed into an algorithmically coherent system.

The same recognition, that architectural problem solving typically relies on previous knowledge is also behind the space languages proposed by Yessios [12] and the shape grammars proposed by Stiny [13] and Gips [14] in the early and mid seventies. Yessios explicitly acknowledged the influence of the pattern languages, while the formalism of both the space languages and the shape grammars was based on Chomsky's work on generative grammars [15].

Not only the recognition of the significance of knowledge and past experiences has entered architectural design research since at least the late sixties, but also much of the formalism, such as production systems, which have today become popular through our acquaintance with the diagnostic expert systems. The point to be made, of course, is that the formalism we "borrow" today is frequently foreign to the nature of the architectural problem, while earlier work was more successful in deriving its basis directly from architectural semantics.

Types of architectural knowledge: While in this section I shall actually propose a few types of knowledge which need to be incorporated in an architectural design oriented CAD system, the main point is that architectural knowledge is much more than logic rules expressed in some symbolic representation. What this knowledge is and ought to be has yet to be sufficiently researched. Let me outline a few types of architectural knowledge as a minimum requirement:

a. Constructive Knowledge: This is the knowledge which allows us to put a building together correctly. It may be considered of a "low" level since it can be effectively incorporated in a straightforward manner and has little need of an inference engine. This is mainly the knowledge upon which most of the automatic procedures of a constructive character are based.

b. Architectural functions: This is the knowledge which tells us how similar problems have been solved in the past; the knowledge which Alexander represented through his pattern languages. The most common representation of this knowledge is through floor plans or functional diagrams.

c. Historical and theoretical background: Beyond its functionalities, architectural design derives inspirations from past histories and styles, aesthetic rules of composition, theories of symbolic signifiers, etc. This type of knowledge being the least factual, is the hardest to incorporate in
Acquisition of Architectural Knowledge: The acquisition of knowledge remains a major research issue. Typically knowledge is being collected by interviewing “experts” of a particular field. But efforts are also underway aimed at the automatic acquisition of knowledge.

The acquisition of knowledge takes a special character in architectural design. The first category outlined, the constructive knowledge, is of a more or less objective character and can be collected through the conventional methods of expert systems. But this is not true for the other types, since past architectural experiences are generally interpreted subjectively by different designers. Furthermore, these past experiences need to be transparent and readable by the user. Therefore, they have to be presented graphically and nobody can expect an architectural designer to be willing to trace symbolic representations.

I suggest that the past architectural experiences, whether they refer to functionalities depicting floorplans or to historical precedents of compositional schemes, should be entered into the computer graphically and through the same sketching procedures provided by a CAD system for the generation of designs. Any internal processing aimed at depicting the knowledge which a past solution contains should be hidden from the user. Whenever this knowledge is required to be presented to the designer, it should be done through the only representation he/she is comfortable with: graphically.

What is envisioned here is a knowledge basis which is first of all a library of prior architectural solutions. The user may request to directly view a number of examples of prior solutions, or he/she may request to be informed about the “wisdom” extracted and inferred from the individual solutions stored in the knowledge basis. This can be done in the form of asking advice while he/she generates his/her own solution or the system may be asked to evaluate and critique a tentative solution already generated by the user.

It should be kept in mind that the above problem solving activities occur from within a CAD system into which the proposed knowledge system is fully integrated. As a matter of fact the same modeling procedures proposed earlier are used to process and utilize accordingly the knowledge which the system incorporates.

Concluding Remarks: CAD in Architectural Schools

Having pointed out that currently available CAD systems are not even close to being the architectural design machines which might prove capable of helping us produce better designs, I then outlined a few areas which can offer the basis for the development of truly design oriented CAD systems. My list of areas which require further research is not exhaustive. I primarily discussed those areas and directions which are currently under investigation at the CAAED Lab of the Ohio State University.

We frequently hear declarations by architectural schools that they do not care to be in the business of producing software. That they simply need to be concerned about making their students computer literate and about exposing them to the “design” tools which are commercially available. Hopefully this presentation has pointed out that such “design” tools are not yet in existence. We are, therefore, forced to use systems which incorporate the engineer’s and the computer programmer’s concept of architectural design.

The main misconception of the above position is that to get involved in the exploration of what a design machine ought to be implies advanced training in computer programming. This is certainly not the case. While it is important for architectural students to have a basic understanding of algorithmic processes and computer programming, architectural schools cannot be expected to produce the highly skilled programmers required for a professional grade implementation of a CAD system. What architectural schools need to produce are strong designers capable of telling the programmers what systems they ought to implement. At OSU, we call it “educating designers of design systems”.

While the position many schools are forced to take is perfectly understandable, since they have to get equipped with CAD fast and they can only do it with what they find available, there is a risk
that this might be established as a philosophical position. If it happens, then the young architects we produce will be exposed to the wrong edge of the double-edged sword and architectural design will suffer. We may certainly be led to the "franchization" of architecture which Mitchell has talked about [1]. On the other hand, should we be able to find ways to expose our students to the right edge of the sword, we may then expect CAD to help us produce architectural quality surpassing any previous achievement. Concluding, I would like to recommend that architectural schools reconsider their position towards CAD. Rather than viewing it as something they have no choice but to live with, they may want to start viewing it as a technology which may lead architecture to new heights of achievement.

References