Get with the Program: Common Fallacies in Critiques of Computer-Aided Architectural Design

Ulrich Flemming

EDRC 48-35-94

Get with the Program: Common Fallacies in Critiques of Computer-Aided Architectural Design*

Ulrich Flemming
Department of Architecture
*and Engineering Design Research Center^*
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract This paper introduces four common fallacies that are often developed explicitly or implied when approaches toward computer-aided architectural design are criticized from a broader "philosophical" perspective. It suggests what the author considers more fruitful directions for research in connection with the issues raised. The first two fallacies are very general. The first of these treats design as a monolithic, indivisible process that cannot be decomposed and thus partially supported. The second one insists that computer aids support current practice as it stands and rejects approaches that challenge that practice. The last two fallacies are very specific. The first of these occurs when shape grammars and related mechanisms are criticized for being based on a 'linguistic analogy." The last one deals with a specific version of appeals to authority that treats Heidegger as the ultimate arbiter in resolving philosophical issues in connection with computer-aided architectural design.

[1] To appear in *Environment and Planning B. Planning and Design*
[2] **The Engineering Design Research Center at Carnegie Mellon University is aiy NSF-supported engineering research center.**

Introduction

As I browse through compilations of papers on computer-aided architectural design, such as conference proceedings, or review drafts that come across my desk becaulse I happen to be on certain editorial boards, I notice an increasing number of papers that take issue with certain approaches toward computer-aided design or question even the possibility of computer-aided design itself on philosophical grounds, where I use the term "philosophical" broadly. My initial excitement about die appearance of such papers, based on my own uneasiness about some of the assumptions that appeared to underlie - consciously or unconsciously - published work in the field, has since given way to disappointment and weariness because more often than not, I found the arguments put forth ill-founded and unconvincing.

I also started to notice recurring arguments, open or implied, that I consider essentially fallacious. I introduce these in tike present paper with the aim to clear the air and to encourage what I consider more interesting and fruitful investigations.

The first two fallacies are very general and appear in many disguises. The last two fallacies are very specific. They all have in common that they misrepresent the value of work in the area of computer-aided architectural design, raise pseudo-issues or divert attention from more fruitful directions. That is, they stand in the way of much heeded progress in the field.

This is not a strictly scholarly paper. I want to stimulate and, at times, amuse readers more than settle issues once and for all. I therefore adopt a more colloquial tone and keep references to a minimum, to occasions where I use direct quotes or refer to specific results. Having decided that some sources should be quoted explicitly, I must also stress that there is more to most of the quoted papers than I am able to take into account in the present context. They may use fallacious arguments sometimes, but they don't do this all the time!

The all-or-nothing fallacy:
If it ain't good for everything, it ain't good for nothing.

Some arguments go like this: "Design is ... (subjective, culturally determined, confronted with ill-defined problems etc.), but computers cannot... (be subjective, take culture into account, deal with ill-defined problems etc.). Therefi^eH^raf3%iters cannot design (or support design in a meaningful way)." I consider, this argumerrt fallacious because it treats design as a monolithic, indivisible process that caQ only be supported *in toto* or not at all. This view would make sense only if thfe cfeigft erf a building were created by a single designer accomplishing everything that has to be done in this process in a single move, which would require, among other things, a short term memory of unlimited capacity and universal expertise on t]pe pirt oi thf designer. In reality, building design consists of a myriad of decisiqiis made by a large number of people representing various disciplines who focus oh.small sets of issues at a time and use a variety of tools. That is, the monolith "design" shatters in reality into a large number of fragments.

The tools designers use have traditionally included pencil and paper, calculators, copiers, references, communication aids etc. None of these tools was expected to support every aspect of design, which nevertheless did not prevent their employment because each supported *some* aspect well enough to make it useful, if not indispensable, when this aspect had to be considered. Computers have; now been added to this array of traditional tools. Why should they have to be able to deal With everything?

The more interesting question is what aspects or subtasks of the design process can benefit from computer support, either because traditional tools have been deficient or because they have not been available in the first place, this type of question did not arise in connection with a tool like the telephone because its use was obvious. Computers, in contrast, can be programmed so that they can support a broader spectrum of tasks the boundaries of which, I claim, have yet to be established. But then, the argument goes, design may be decomposed in practice, but the decompositions are flexible and evolve dynamically so that a finite suite of tools can never support a potentially infinite number task formations? This brings us back to the start why should computers be required to support everything? Isn't it good enough to start with tasks that occur frequently, that may benefit from computer support and see how they can be supported? Yes, this may require starting with an initial selection of tasks and some initial task boundaries. But programs go through versions, where later versions take the experience with prior versions into account. We also tend to underestimate the potential of computers to adapt themselves to evolving situations; the section on shape grammars will return to this.

If we ask which tasks can benefit from computer support, some immediate criteria come to mind: tasks that can be done better or faster by computers, or tasks that can only be done by computers. An example for the former is the calculation of stresses in a truss; an example for the latter is the simulation of energy usage for a schematic building configuration over an entire year. A counterexample is the leap of the imagination that lead LeCorbusier to draw the initial silhouette of the chapel at Ronchamp or made him conceive of the general circulation diagram for the Visual Arts Center at Harvard while observing students filing through the snow. Using the criteria above, even if computers could do anything like this, would we be sure that they could do it better or faster than human designers?

This is not to say that computers may not be able to stimulate the imagination of designers, for example, by bringing up pictures of the context surrounding a site or based oitlmalogies found in these images. When I speak of computer *support* for tasks, I specifically include this type of use. That is, I don't imply that the computer executes the task automatically and autonomously; it may be part of a cooperative arrangement where people and machines perform different functions.

The spectrum of tasks between the extremes, tasks that we know computers can support successfully and tasks that they cannot support successfully, is open-ended, not so much because we have no exhaustive list of tasks performed by designers, which could be checked one-by-one if they could be supported, partially or entirely, by computers, but because we do not yet know the limitations of computers in supporting design tasks and, as we try to push the boundaries of their capabilities, new tasks are likely to emerge.

One reason given by people who fear decomposing design taste in the context of computer-aided design is that this isolates or decontextualizes information and its use. I wonder if this argument hoick up under scrutiny. Take my word processor as an example. It is true, it contains a dictionary that is superficially more isolated from its context than the dictionary I have on my desk. But it is the dictionary in the computer that I use, not only to check the spelling of a particular word, but in an entire paper for typos and words that I misspelled because of my limited knowledge of English orthography. When the program checks the document, it is certainly as contextual as the dictionary on my desk when I use it, and my interaction with the computer is certainly more intricate because the computer comes up with observations and suggestions on its own, to which I may respond by either accepting or discarding them (and laughing about some of the suggestions it has made, such as change "Heidegger" to "headgear"). That is to say, a tool has to be viewed in the context of its use, and the entire organizational unit performing a task has to be evaluated for its effectiveness in executing the task: a pencil without a designer is incapable of producing a drawing, but so is a designer without a pencil; it is only when we put the two together that things start to happen!

Conversely, tools have to be designed to be useful in their intended context Take spelling checkers again as an example: those that are programmed poorly so that they flag the same proper name repeatedly in the same document even after I have accepted it once annoy me decidedly more than those that are programmed to accept my decisions graciously. But none of the spelling checkers I have used would flag "principle" when I should have used "principal;" that is, they do exactly this, check spelling, and cannot deal with usage issues. It would be as pointless to blame them for this as it would be to blame a horse for not being a cow (even if one needs milk badly). At the same time, it would certainly be interesting to find out if spelling checkers could be expanded to take usage into account.

All of this reaffirms what we know about tools anyway: they can be used properly and they can be used improperly, they can be designed better or worse for their intended use, and the ultimate responsibility for using a tool at all and for using it in specific ways rests with the user. I submit that if we view computers as tools rather tha autonomous agents, many of the problems raised in philosophical critiques of computer-aided architectural design will simply go away. But it is also clear that selecting promising tasks and developing computer-based support tools are not enough. The way in which people learn to use a program and %ow it makes its capabilities available and interacts with its users are crucial ingredients infethe ultimate success of the program.

A response to the examples that I have cited may be: yes, compute can do this well, but this is not truly (part of) design. I strongly warn against adopting this attitude. We hear with increasing frequency that architecture as a precession is becoming marginal precisely because architects have given up mordkjmd more responsibilities for aspects of building design that occupants and cliefts are truly interested in: environmental comfort and control, ease of access, complefaon on time and budget etc., aspects that may be more readily controlled if cdJ^Duters are employed. Do we really want to use the computer as a pretext for the further reduction of responsibilities architects consider worthy of their attention?

A weaker version of this response accepts the usefulness of computers, but underplays their impact because, it is claimed/most tasks in design are ill-defined and thus not amenable to computer support. I heard Horst Rittel in a lecture compare the well-defined design tasks to a few raisins in a cake of ill-defined problems. I have the opposite impression: the ill-defined tasks are the raisins, which are, in addition, clustered at the top and almost absent at the bottom. Neither of us has provided any evidence for his claim!

The principle researchers in the area of computer-aided design who are interested in creating systems for practical applications should accept is that they must account for themselves and their audience what aspect of design they want to support and what benefits they expect. Based on this, the success of their work can be evaluated, especially by the intended users, and iterative improvements can be made. Researchers should also be ready to accept that their expectations have been disappointed.

I no case should they replace the "all-or-nothing" fallacy with the "something-is-everything" fallacy by declaring "design is ... (search, problem-solving, goal-oriented etc.)" and, based on such a suitably reductionist definition of design, conclude that their program does indeed support every aspect of it.

Dr. Pangloss[1] fallacy:
This world is the best ofall possible worlds.[3]

There are researchers who concede that computers may have a place in building design, but insist, explicitly or implicitly, that computers support this process *in its present form.* Researchers with a background in Artificial Intelligence (AI) seem particularly susceptible to this view. I have argued elsewhere that this may be due to the fact that AI researchers have traditionally addressed "common sense tasks," such as vision or natural language understanding. These researchers may be so impressed by the feats people perform in these tasks and by the difficulties they encounter when they try to program computers to do the same that they perceive programs that perform like people as the ultimate goal of their endeavors (Hemming, 1994). But if one accepts the principles established in the preceding section, namely that computers should be viewed as complementary to people in the design process and given tasks or functions they perform better than people, one will come to the opposite conclusion: the goal of computer-aided design is to create programs that work very differently from people because if they work exactly like people in a specific capacity, they will not be able to do better than peoplq in this capacity.

Take for example the proposal made in (Papazian, 1993). The author observes that designers work in "see-move-see[11] cycles (as opposed to the generate-test-

[3] Dr. Pangloss is a character in Voltaire's *Candide.*

regenerate cycles employed by many computer programs), where the focus of design shifts in every cycle depending on the specific opportunities perceived, and concludes that computers should support more directly this type of process. Let's assume for a moment that the underlying observations are valid (a very big assumption given the scarcity of empirical data supplied!). Does this imply that computers should support and strengthen the observed practice? I think we have to be careful with such conclusions. We have witnessed grotesque failures of buildings, especially ones that are highly praised in the journals, in their first years of occupancy, leading to additional costs in the million range and extended law suits (the State of Illinois Center in Chicago and the High Museum in Atlanta come readily to mind). Could it not be that these failures are precisely the result of opportunistically shifting fod that prevent designers from ever reaching a view that would take the entire spectrum of relevant concerns into account? If people are good at seeing-moving-seeing, wouldn't a program that reminds designers of things they may have overlooked be more useful in this context? After all, there is a more cynical explanation why designers work in see-move-see cycles: their short-term memory is extremely limited, and they couldn't adopt a more holistic view even if their life depended on it.

Take communication as another example. We are told that

> with introduction of knowledge-based and expert systems the nature of communication flow has changed. The tradition of draughting and its conventions developed to facilitate the interaction of various members of the community which structured the building process: client or patron, architect, draughtsman, builder. These players formed a cultural web mediated by a commonly held representational vocabulary, which allowed each to bring their [ski] own knowledge and interpretation to bear on the project (Calvo, 1993, p. 158).

Now just about everybody who studied the drawing-based communication between the participants in the building process agrees that it is in a sorry state. Horror stories abound about information being lost, communicated with errors or too late, and misunderstood between the players involved, which include all consultants. The results are significant cost overruns due to change orders and construction delays or faulty execution. Do we really want to hold this practice up as an ideal that has to be preserved?

It is true that computers may actually exacerbate existing problems, for example in the area of communication, where programs based on incompatible data formats, if not conceptualizations of design, have created additional communication barriers. But the area of distributed collaboration that is emerging as a strong focus of research in computer-aided design addresses precisely theses issues; it has re*established especially the need for a shared vocabulary that is open to interpretations (see, for example, Gruber et al. 1992).

It is interesting to observe that this work has brought up a severe shortcoming in traditional drawing-based communication, namely the loss of *desigtijntent* or *rationale,* which leads to misinterpretations and design changes with unanticipated side effects. A significant problem addressed by work in distributed collaboration is precisely the communication of intent (Klein, 1992). Research in computer-aided design that challenges established practice is potentially so exciting because it motivates us to reexamine every aspect of that practice and look at it with fresh eyes.

Thus, I consider the aim to correct observed shortcomings in current practice a very good motivation for work in computer-aided design. Having said this, I must stress that I am not falling into the trap of the technocratic fallacy that assumes that non-technical problems have technical solutions. I believe that such problems as are caused by bad communication practices demand responses that go far beyond what computers can accomplish. They demand that the participants in the building process reconsider how they perceive their role and responsibilities in that process, how they organize their interactions, and - very importantly - how they educate the next generations of students. My point is that we should not become nostalgic about the current state-of-affairs in building design and assume, implicitly or explicitly, that it is to be perpetuated when new tools are introduced into that process or, conversely, reject attempts to create such tools solely because "that's not how designers do it"

At the same time, we should not assume that *every* aspect of current practice can improve through computer support. Methods that work well as they are, or fall principally outside the tasks computers may handle, should be left alone. The initial conceptualization in the building process mentioned in the preceding section is an example. Another example is design-by-sketching, which is prominent particularly in the early design stages and characterized by the liberal use of transparent paper. I consider attempts to transport this process *literally* to computers, that is, turn them into a *sketching* tool, wrong-headed because I can't imagine how this could improve upon the current process, especially upon its speed and ease of execution.

If we accept, on the other hand, that designers sketch because they want to explore rapidly the possibilities and opportunities that arise in a specific design context, a very interesting motivation for experiments with computers emerges: can computers offer *alternative* means for the exploration of possibilities that can be used in parallel with sketching or, if they prove superior, may replace it at times? More generally, we should not remain fixated on or make a fetish of design-by-drawing, which became possible only with easy access to virtually unlimited amounts of paper and represents thus a very recent development in the history of building design.

This brings me back to the principle established in the preceding section: researchers in the area of computer support for architectural design, especially in academia, should be able to identify clearly to themselves and their audience the aspect of the building process they address and why they hope that they can improve upon current practice. Furthermore, they have to take work in the rapidly expanding field of human/computer interaction into account, which stresses the need to consider the ultimate users of the system under development from the beginning, to test the evolving prototype with these users, and to be ready for iterative improvements. Finally, they have to accept the risk that their ideas are not, or not immediately, accepted by the design community. But research wouldn't be worth its name if it wouldn't take such risks.

Quoting out of context:
People who use the same word mean the same thing.

George Stiny has told me that he keeps his own private treasure box where he collects stupid statements that have been made about shape grammars. I have noticed myself for a long time how frequently this mechanism is misunderstood, misrepresented, or criticized for the wrong reasons*

The very term "shape" is clearly defined in Stiny's papers as a collection of "maximal" geometric entities, like line segments, and therefore cannot be substituted, when one talks about shape grammars in Stiny's sense, for anything that "shape" may mean in common usage. Furthermore, the rules or schemata that work on such shapes have to be understood in the way in which they are defined, specifically as mechanisms capable of finding "emergent" shapes, that is, combining portions of maximal entities (of which there may be infinitely many) in novel and unexpected ways. This crucial point is so frequently missed that I have adopted the habit of bringing this immediately to the notice of people who tell me that they are going to study shape grammars. Granted, the idea that a well-defined computational mechanism is able to interpret an equally well-defined finite collection of elements in infinite many ways is so unexpected that people may not be prepared to register this even if it is stated clearly and illustrated with examples. As a result, the literature dealing with shape grammars is riddled with mistakes.

But this paper is not about bad scholarship in general, but about fallacies, and I mention shape grammars because a specific fallacy is particularly frequently committed when shape grammars are criticized on philosophical grounds: it is alleged that the use of shape grammars and similar mechanisms is based on a "linguistic analogy" and that such analogies are not permissible when we deal with the compositional or generative principles at work in fields like architectural design.

I am convinced that languages as means of communication evolve differently from architecture and that the conventions that connect words with meaning and the way in which these conventions are learned have no equivalent in architecture (see for example Scruton, 1979, ch. 7, for an extensive critique of the linguistic analogy in architecture). But shape grammars and related rule-based mechanisms have *not* been applied to architecture based on a linguistic analogy, which should be clear to everyone who reads the original literature carefully. Shape grammars have their origin in Post's production systems (Post, 1943). They were adopted by Chomsky in the form of "phrase structure grammars" to investigate problems in linguistics (Chomsky, 1957). The name "grammar[11] has since then stuck to these mechanisms, even when they were used in areas far remote from linguistics (see Gips and Stiny, 1980, for a review of this work). That is, the "grammar" part in shape grammars is to be understood in a purely technical sense; no analogies, legitimate or not, are implied.

Readers who are still confused may consider this: Mandelbrot's fractal generators, in their pure form at least, are a subset of shape grammars (Stiny, 1977). If shape grammars are based on a linguistic analogy, so is fractal geometry: a blatantly absurd conclusion!

I have been at pains to point this out in papers on shape grammars and related mechanisms intended for non-specialists (e.g. in Hemming, 1987). To no avail: to the present day, an enormous amount of paper is being wasted (see for example Fleisher, 1992) on refuting the linguistic analogy allegedly implied by people using shape grammars. It is true that the analogy is sometimes stated as a justification for the use of grammars in design, for example, in (Coyne et aL 1990); but a careful reading of the original literature would clarify misconceptions that may arise from this.

However, given the prevalence of such misunderstandings, should we not use a different term for the type of mechanism in question? I'm not sure about this because I suspect that this would create a new round of misunderstandings, both because the new term might be misunderstood in its own right and because the connection with the tradition in which this work stands would be lost It is a fact of life that at any time, there are too many meanings chasing too few words in the dictionary, hence the overloading of words with meanings and the necessity to understand the meaning of words in their context. When an art dealer, a grammarian, and a student of programming languages talk about an "object," they refer to very different things. It is an indication of the low standards in the literature on computer-aided architectural design that something so painfully obvious has to be said at all!

This is not to say that the selection of a word like "object" in the above contexts is totally arbitrary; it refers in each case to an entity that can be recognized as such and distinguished from other such entities. I'll get back to this in connection with grammars.

In my own work, I have been able to use shape grammars, for example, to provide "constructive" explanations for rather irregular compositions like the American Queen Anne house (Flemming 87a) by developing a set of "rules" that generate them step-by-step, where each step is based on an identifiable technical or compositional logic very much in the way architects use sequential diagrams to explain the logic behind a certain composition. These rules, taken together, provide explanations that are at once clearer and more comprehensive than the static descriptions given usually for architectural compositions. The rules themselves were developed in the course of many experiments in which different rule sets were tested with the help of a computer, which turned into a laboratory for experimentation with forms and formal principles. This was a truly exhilarating experience and suggested to me a strong motivation for using computers in so-called creative work: to provide? a mirror that throws my own ideas back at me under multiple perspectives.

A crucial aspect of this work was, of course, that I wrote my own rules; that is, I tried to improve the program through several versions until it satisfied my intentions. This experience is less removed from practice than it may appear. Among my colleagues that maintain an active architectural practice, two are particularly happy with the CAD system they use in their office because they customized their respective system using a feature that has become standard for commercial CAD systems, an application programming interface (API) that allows for adaptations and links to other programs. To my surprise, the possibility of customization is almost universally overlooked when computer-aided architectural design is criticized from a philosophical perspective.

My own work with shape grammars and related mechanisms has demonstrated to me that they offer, in principle, a particularly rich palette of customization possibilities (I say "in principle" because the form of interaction in which these possibilities can be best utilized has still to be found). I would also like to point out that the ability of shape grammars to discover emerging shapes and to reinterpret representations shows that computational mechanisms can be more flexible in their response to evolving design descriptions than common wisdom has it It is for these reasons that I am so concerned with misrepresentations of these mechanisms.

The interesting question about grammars is not whether they should be applied or not in architectural design, but why they work so well *when* they work welL This brings me back to the linguistic analogy. It is remarkable that it continues to surface, not in work on shape grammars, but when architects look for a metaphor to express what they sense intuitively (F. L. Wright is an example) orwhen historians investigate certain aspects of architecture. I find, for example, the use of the language analogy in the following passage very evocative:

> The slow elaboration of architectural materials inherited from antiquity into a formal canon has several times been linked to the development of a vocabulary, a grammar, and a syntax of architecture. This analogy is particularly tempting when one looks down a classical street where a multitude of town houses in shifting modes and changing rhythms relate to each other in a civilized manner, like persons engaged in a protocolar official manner or in a polite conversational occasion sprinkled with pleasantries. As in a string of words when people are chatting, there is endless "variety/[1] "accident" together with "predictable repetitions" and turns of phrases inserted out of courtesy in the midst of an argument or the transmission of information. This wonderful universe is made possible only by some kind of underlying normative apparatus that controls the entrance of elements into the discourse (Tzonis and Levaifre,212)

Clearly, the analogy emerges independently in different contexts; in fact, it appears irrepressible. This may be the reflection of a similarity between architecture and language that is given not by a direct analogy, but by similarities at a deeper level. Languages provide the means to construct sentences that can be interpreted because they follow recognizable rules which can be expressed in a grammar. Architectural compositions are what they are because they can be recognized as such, that is, as non-arbitrary configurations of pieces that are non-arbitrary precisely because we can identify the underlying "grammar," if only *a posteriori;* Scruton demonstrates this very convincingly when he introduces the notion of *intentional regularity* in architectural compositions (loc. cit). That is to say, similar notions may be useful in the study of languages and in the study of architecture because the two enterprises share a more abstract aim, to construct *recognizable patterns,* albeit certainly by different means and with different intentions. This may be the reason why recursive rule systems emerge independently as useful representations and experimental tools when such patterns are under investigation, be they fractals, architectural compositions, or declarative sentences.

Further studies along these lines may prove illuminating. For example, an investigation of the different means by which recognition is achieved in language and in architecture may lead to a study of the role of convention and explicit instruction in the case of languages and to a study of the role of tradition, captured in the notion of "style," in the case of architecture. *This* is the type of philosophical critique that I would consider fruitful in connection with grammars and related mechanisms in architecture.

Appeal to authority: *In Heidegger we trust!*

In this section, I will deal briefly with sterile applications of a variant of the *modus ponens* that goes like this:

Authority *A* says that *a* implies *b*.
We observe a.
***Ergo,* we assert *b*.**

I call an application of this mode of argument sterile when it fails to do at least one of die following: (i) establish why we should believe *A* in the first place; or (ii) establish that *a* is, in fact, true- The authority I am concerned with in particular is Heidegger, who has been used, *inter alia,* to criticize certain approaches underlying computer-aided design and thus bring "poststfucturalist" writing, finally, to our field (see for example Coyne and Snodgrass, 1993). Better late than never? I'm not sure: one can make progress sometimes simply by avoiding a detour!

Let me start with (i). The studies I have in mind present Heidegger always as an authority in no need of justification; he enters the stage as *deus ex machina* to rescue hapless researchers in the field of computer-aided design from the clutches of the infamous Descartes and assorted other villains (like AI researchers). There is not even a hint that his ontology has been under severe attack for a long time, not so much because of his open support for the Nazis (which could be dismissed as an *ad hominem* argument), but on its own grounds, which then *explains,* for example, why he was susceptible to fascist ideology (see Sheehan, 1993)- It is absolutely necessary that, if someone as controversial as Heidegger is introduced as an authority, a careful argument be presented why he should be treated as such, taking the critical appraisals into account

But let's assume that Heidegger's (or anyone else's) authority has been established. We still have to make sure that our observations about *a* are, in fact, correct. And here I must confess that the characterizations of computer-aided architectural design that I find in the papers in question of ten unrecognizable when I compare them with my own experience or that of my colleagues in this field. For example, I find assertions that a program is based on a computational model of *design* when, in fact, it only attempts to provide computational support for an *aspect* of design or for a specific *subtask,* which, in addition, may be tackled only in close cooperation between designer and computer; that is, it may not even be based on a

computational model of a single subtask of design; in fact, it may not be based on a computational model of anything. I am also not alone in such observations; it has been argued that a Heidegger-based critique of formal grammars in design misses the point entirely because this work satisfies the very criteria it allegedly violates (Woodbury, 1993).

In its most degenerate form, the argument discussed here introduces the observations under review *already in Heideggerian terms* (that is why I reversed the usual order of steps 1 and 2 in the modus ponens above). This makes the conclusion automatic and the whole argument circular

> Heidegger says the separation between subject and object is bad.
> System *x* separates between subject and object.
> Therefore, system *x* is bad*

Even arguments that stay well above this level (like the article cited above) expose themselves to the danger of circularity when they start, not with particular observations, but with philosophical theories, which clouds the perception of subsequent observations on the part of the reader, if not the author. That is, they risk what they explicitly try to avoid, scholarship founded "upon the development of theories against which we measure our experience" (Coyne and Snodgrass, lit tit, 102).

A more convincing start would be provided by the observations of the people who experienced the problems under investigation (and who are unlikely to come up with statements like "this program separates subject and object, and I have problems with this") *in their own words.* Of course, these statements will reflect biases, but these are the biases we should be interested in; for hermeneutical cyclists, they establish the very *terrain* in which they should be happy and eager to circle! Authorities (preferably not Heidegger!) can then be brought in to shed light on the situation and, possibly, suggest remedies. Implementing these can have, in prindple, two outcomes: the situation does or does not improve. In the latter case, our authority may lose some of this very authority. That is, whenever we use theories to interfere in empirical situations, we should be ready for one of two outcomes: the situation improves, or our theory is disproved, and both results are interesting. I know, of course, that complex situations normally do not lead to such unambiguous results. I simplify in order to establish a principle: if you don't want to measure experience against theories, you should be ready to measure theories against experience.

The irony is that Heidegger may not be needed after all. For example, Coyne and Snodgrass conclude, in the quoted article, after a lengthy philosophical discussion with a list of desired characteristics of a CAD system that takes their concerns into account. It so happens that this list contains many of the objectives pursued by the members of my research team. But we arrived at these not top-down via Heidegger, but bottom-up from our interest in developing a system that would be accepted by and useful to designers.

For a more light-hearted introduction to Heidegger's critics and his commercialization, for example, when his jargon is used to dress up common sense insights or to disguise "crashing platitudes," I recommend (Gottlieb, 1990).

References

C. M. Calvo (1993) "Some epistemological concerns regarding artificial intelligence and knowledge-based approaches to architectural design - a renewed agenda," *Education and Practice: The Critical Interface (Proc. ACADIA 1993),* F. Morgan and R. W. Pohlman, eds., n. p.,.155-162

R- D. Coyne, M. A. Roseman, A. D. Radford, M. Balachandran, J. S. Gero (1987) *Knowledge-Based Design Systems,* New York: Addison-Wesley

R. Coyne and A. Snodgrass (1993) "Rescuing CAD from rationalism" Design *Studies,* vol. 14,100-123

N. Chomsky (1957) *Syntactic Structures,* The Hague: Mouton

A. Fleisher (1992) "grammatical architecture?" *Environment and Planning B. Planning and Design* , vol. 19, pp. 221-226

U. Hemming (1987) "Syntactic structures in architecture: teaching composition with computer assistance," in *The Electronic Design Studio,* M. McCullough, W. J. Mitchell and P. Purcell eds., Cambridge, MA: MIT Press, pp. 31-48

U. Hemming (1987a) "More than the sum of parts: the grammar of Queen Anne houses," *Environment and Planning B. Planning and* Design , vol. 14, pp. 323-350

U. Hemming (1994), "Artificial intelligence in design: a mid-term review" in *Knowledge-Based Computer-Aided Architectural Design* G. Carrara and Y. E. Kalay eds., Amsterdam: Elsevier, pp. 1-24

J. Gips and G. Stiny (1980) "Production systems and grammars: a uniform characterization/^{1} *Environment and Planning B,* vol. 7, pp. 399-408

A. Gottlieb (1990) "Heidegger for fun and profit" *New York Times Review of Books,* Jan. 7

T. R. Gruber, J. M. Tenenbaum, J. C. Weber (1992) 'Toward a knowledge medium for collaborative product development" in *Artificial Intelligence in Design ^{S}92,* J. Gero (ed.), Boston: Kluwer Academic Publishers, 413-432

M. Klein (1992) "DRCS: An integrated system for capture of designs and their rationale" in *Artificial Intelligence in Design ^{K}92,* J. Gero (ed.), Boston: Kluwer Academic Publishers, 393-412

P. Papazian (1993), "Incommensurability of criteria and focus in design generation," *CAAD Futures* '93 *(Proc. of the Fifth International Conference, on Computer-Aided Design Futures, Pittsburgh, PA),* U. Flemming and S. Van Wyk eds., Amsterdam: Elsevier, pp. 111-125

E. L. Post (1943), "Formal reductions of the general combinatorial decision problem," *American Journal of Mathematics,* vol. 65, pp. 197*215

R. Scruton (1979) *The Aesthetics of Architecture* , Princeton, NJ: Princeton University Press

T. Sheehan (1993) "A normal Nazi" *The New York Review of Books* vol. 40 no. 142 (Jan 14), 30-35

G. Stiny (1977) Review of *Fractals: Form, Chance, and Dimension* by B. B. Mandelbrot (1977), *Environment and Planning B,* vol. 4, pp. 248-9

G. Stiny (1981) "Introduction to shape and shape grammars," *Environment and Planning B,* vol. 8, pp. 343-351

A. Tzonis and L. Lefaivre (1986) *Classical Architecture. The Poetics of Order,* Cambridge, Mass.: MIT Press

R. Woodbury (1993) "Grammatical hermeneutics" *Arch. Science Review,* vol. 36 (no. 2), pp. 53-64