Deconstructing AutoCAD

Toward a Critical Theory of Software (in) Design

Mahesh Senagala

School of Architecture, University of Texas at San Antonio

www.mahesh.org

AutoCAD maintains a nearly 70% market share in the PC-based AEC sector and wields enormous influence over design and production processes in architectural firms and schools. Such an impact is, perhaps, more than what a single building can hope to achieve. The design implications of such a market monopoly are many. Based on Derridean operations of deconstruction, the paper will deconstruct AutoCAD's latent agenda. The paper will do a critical close reading of AutoCAD for its design preferences, spatial conceptions, worldviews, resistances, stratifications and organizational predispositions with respect to architectural design process. For purposes of brevity, this paper will focus on the architecture of AutoCAD's interface. The results of the paper would be the *beginning* of a critical theory that can be employed in the process of software design for design professions.

AutoCAD, deconstruction, critical theory, software design

Introduction

AutoCAD maintains a nearly 70% market share in the PC-based AEC sector and wields enormous influence over design and production processes in architectural firms and schools. Such an impact is, perhaps, more than what a single building can hope to achieve. The design implications of such a market monopoly are many. The paper intends to fill the void created by the paucity of critical work on this topic which affects millions of architects, students and building users. While texts, buildings and even movies have been deconstructed widely, software systems have not been subjected to the same degree of systematic critical "*solicitation*" despite the deeper impact of the software packages. Also, while the field of architectural design has been greatly supported by a wide body of theoretical knowledge and critical studies, sadly enough there are no similar discourses about software design available to us. Based on Derridean operations of deconstruction, the paper will deconstruct AutoCAD to reveal its surprising latent agenda. The paper will be a critical close reading of AutoCAD for its design preferences, spatial conceptions, worldviews, resistances, stratifications and organizational predispositions with respect to architectural design.

For purposes of brevity, this paper will focus on the architecture of AutoCAD's interface. *The results of the paper would be the beginning of a critical theory that can be employed in the process of software design for design professions*. The paper will question a number of otherwise taken-for-granted performative/prosaic approaches to and notions about the design of computer-aided design software tools.

Deconstruction of (de)faults, A Solicitation

Structure then can be methodically threatened in order to be comprehended more clearly and to reveal not only its supports but also the secret place in which it is neither construction nor ruin but lability. This operation is called (from the Latin) soliciting. In other words, shaking in a way related to the whole (Jacques Derrida, *Writing and Difference.)*

AutoCAD maybe the first software to unwittingly employ deconstructivist operations to subvert the processes and privileges of architectural design process! AutoCAD cannot just be judged as good or bad. It cannot just be termed efficient or inefficient. It cannot just be analyzed for its "features" or capabilities. It is not the author's intention to write yet another "review" of AutoCAD. AutoCAD presents a strange challenge to architects. The author contends that it is Derridean in the way it functions. We could possibly go the extent of saying that AutoCAD actually employs (inadvertently, I might add) Derridean strategies to question the binary oppositions, assumed boundaries and processes in the field of architecture. *As unlikely and incredible as it might sound, AutoCAD deftly deconstructs the architectural design process.*

Adopting a post-structural approach, the paper will solicit the “(de)faults” of the software as a way of “reading into” the intentions and processes behind the software. Defaults are also a way of understanding the underlying assumptions, fault lines, and the “unconscious” of the software.

Why Deconstruction? Why now?

Why use the much maligned deconstructivist reading now? Why use it after the notoriety that it had gained during the eighties? Why use an “out-of-fashion” device of criticism now?

Michael Benedikt had an answer ten years ago to these questions when he had used the Derridean reading to critique Louis Kahn’s Kimbell (Benedikt, 1991). He was one of the few architects to have actually read Derrida to reveal the extraordinary and latent schemata of the Kimbell. He wrote: “Derrida’s Deconstruction can mean more to architects (and artists) than a transitory aesthetic or a style, and it should not be allowed to devolve into the esoteric, promotional patter and stylish nihilism that it threatens to do.” Further, he pointed out that “Deconstruction’s destiny, I believe, like system theory’s, is to continue to be absorbed into routine intellectual, critical and even scientific discourse.”

I agree with Benedikt about the immense value and profundity of Derrida’s contributions that should not be forgotten as a fad. The baby should not be thrown out with the bath water of stylistic fads. Derrida gives us the critical contraptions that can be employed to systematically dissect a given construct, be it a text or a building or a software system, and we would be well advised to keep the baby, as it were, less the bath water.

Derridean Operations: A Brief Outline

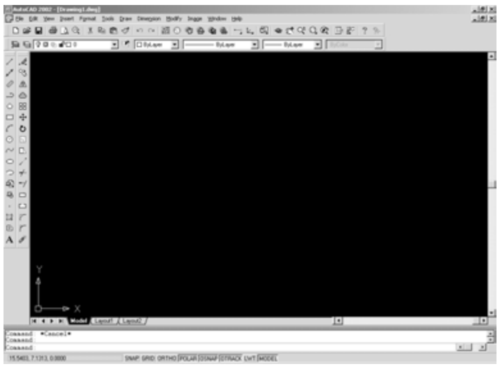
Derrida is known for his inimitable style, critical attitude and politically subversive mode of writing as a means to expose the binary oppositions and reveal the fictitious boundaries of the philosophical texts. Derrida’s operations are not critiques; a critique requires one to stand outside of the subject. Instead Derrida reads those texts from inside (just as a virus invades the host) and exposes the carefully constructed binary oppositions such as good/bad, being/non-being, presence/absence, God/man, writing/speech etc. He shows how the first term is privileged and promoted by suppressing and playing against the second term. In architecture, the binary oppositions usually manifest as 3D/2D, drawing/writing, scale/scaleless, presence/absence, inside/outside, up/down, space/form, and so on. AutoCAD, as you will see, subverts these polarities in a Derridean manner. To me, it is less important to dwell on the question of whether the software creators intended this subversion or not.

What we do understand from Derrida is that philosophical texts employ the following procedure in constructing their textual edifices: Create strong polarities and distance one from the other (speech/writing); privilege the first one (speech is good); quell the second one (writing is bad); ignore the in-between and the undecidable issues

Default: Nothing/Void/Black/Blank

What is the conception of space and dimensionality of the black and almost blank void that you are faced with when you open AutoCAD?

Architects privilege scale and the use scaled grid(s), which AutoCAD’s default black screen subverts. The traditional process of design involves a piece of paper or cardboard, a scale and pens or pencils with which to sketch. But in AutoCAD, by default, one is faced with a black and blank screen. Conceptually it is a black hole, a *niger tabula rasa*. What is the size of the space on that black screen? What tells you



```
Command: *Cancel*
Command:
Command:
```

whether the space is millimeters or kilometers in magnitude? What scale is it at? What plane are you looking at? How deep is it?

What are its boundaries and where are they? These questions cannot be answered without some active intervention by the user. However, even the most avid users remain disoriented and the scale remains undecidable. The black screen essentially stumps any assumptions of the designers about scale or size or height or depth in a truly deconstructivist manner.

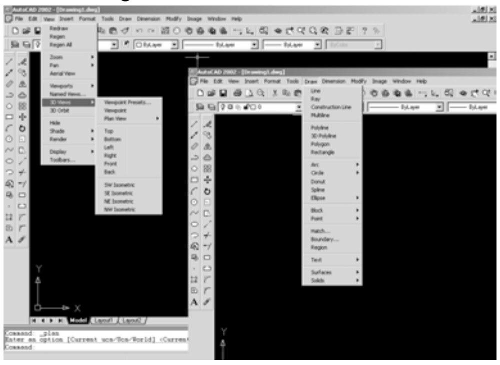
Figure 1: Nothing/Void/Black/Blank

Default: 3D/2D

Architects take pride in their 3 dimensional thinking. Their processes often involve extensive perspective sketching or 3D model making during early design exploration phase as well as during design development and presentation phases. Not surprisingly, 2D becomes a polar opposite and is often looked down upon by designers. AutoCAD effectively exposes the designers' privileging of 3D by employing a number of strategies:

The default screen in AutoCAD is two dimensional (X-Y). It suppresses the third dimension (Z) or the fourth (t); to flip to a three dimensional view, one has to go through ritual involving three levels of menus or obscure and deeply buried icons. Even then one can only arrive at an isometric view, and not a perspective view; generating a perspective view in AutoCAD is a laborious task and discourages extensive use of 3D perspective visualization as a consequence; modeling in 3D in AutoCAD is an awkward, non-native and cumbersome process.

Once again, AutoCAD does a Derrida. It exposes the binary polarities of 3D versus 2D by reversing the polarity and forcing the 2D upon the designer. A close examination of AutoCAD's co-founder John Walker's software development logs throw light on this observation: "A three-dimensional capability is desirable. It appears that an ``extrusion'' feature might be relatively simple to implement and sufficient for some users. Could be an extra-cost option. 2.1." His logs dated September 5, 1983 also make it clear that 3D was low on AutoCAD's priorities during its development. 3D capability was added only in 1985 in Release 6 (version 2.1):



```
Command: _plan
Enter an option [Current ucs/Ucs/World] <Current
Command:
```

Figure 2: Flatland

"Many of our competitors (MCS, Nelson Johnson, ESC), have or will be introducing 3D packages around the time of COMDEX. If we do not have a credible response to queries about 3D, we may be in trouble selling our package. While all drafting is 2D, and almost all users will spend all their time with AutoCAD working in 2D mode, 3D is important more from a marketing perception standpoint than a technical one."

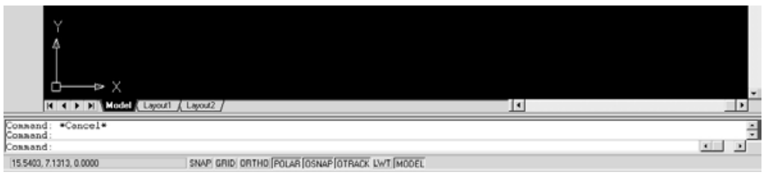
Default: Order/Disorder

Architects define their role as designers of space. They essentially order space in various ways to achieve the performatory or other design goals. Except for a select few, the majority of the architects reject any kind of messiness or disorder. They would like to be able to explain how the entrance leads to the lobby leads to the auditorium, etc. Spatial contiguity, spatial coding and proximity studies form the core of ordering any space, including a software interface.

Now look at AutoCAD's interface (Figure 1). There is a Derridean drive to obfuscate any clear cut organization of tasks, icons and zones. Drawing tools can be found strewn among the icons to the left, among the pull down menus at the top. Window management, layer organization, text management, 3D drawing, viewing, preferences, image management etc. are all dispersed all over the interface in what seems to be a deconstructivist effort to defy a singular comprehension of the interface. There is essentially no rule by which one can find a particular feature geographically on the screen. This fact flies in the face of what architects have always known: clear cut spatial/geographical organization. Thus, yet another binary opposition is reversed by AutoCAD's interface through the strategies of *undecidability* and "*overturning the binarism*".

Default: Drawing/Writing

Architects and other designers take pride in their drawing skills. Often the designers are heard promoting how graphic communication is essential to the design process. Also, designers are often people of few words. If a designer wants to draw a line, s/he draws a line. If s/he wants to draw a wall, s/he draws a wall. We do not see designers having to first write the word "line," write a verbal description of its color, length and other properties before s/he commits to actually drawing it on paper! Drawing is the privileged/valorized mode of communication and expression among designers.



```
Command: *Cancel*
Command:
Command:
15.5403, 7.1313, 0.0000    SNAP GRID ORTHO POLAR OSNAP OTRACK LWT MODEL
```

Figure 3: "A Command Line Runs Through It"

AutoCAD deconstructs that process by introducing writing as a means to drawing. Command line window is the second most prominent element on the interface. In the tripartite division of AutoCAD's interface, command line window forms the "base." Command line is where one essentially writes one's way through a drawing. If one is skillful enough, one does not have to even touch the mouse or click on the screen, ever. By placing a great emphasis on writing and by placing the command line prominently on the geography of the interface, AutoCAD inverts the binary opposition of drawing/writing into writing/drawing, thus challenging and questioning the traditional polarities of architecture.

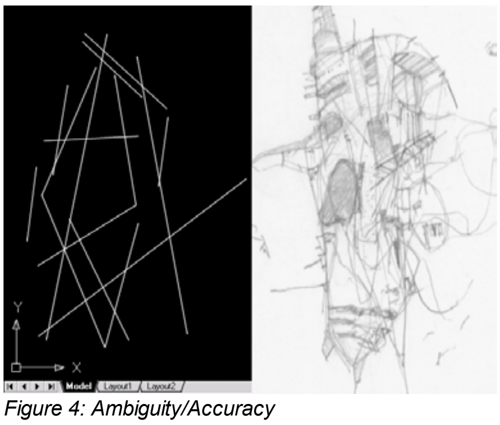


Figure 4: Ambiguity/Accuracy

Default: Ambiguity/Accuracy

Architects prefer to maintain a certain level of ambiguity, over-sketching and looseness of lines in order to avoid premature fixation of form and order, particularly during the formative stages and partially during the design development stage. Sketches, schematic drawings and other exploratory drawings clearly show this quality preferred and privileged by architects: grey marks are made against off white or translucent faded yellow butter paper.

AutoCAD thwarts any ambiguity with its ruthless accuracy (or at least the appearance of ruthless accuracy): pure white lines against pure black background. That is a pure binary reversal. Architects might find its accuracy menacing. But the value of this overturning of ambiguity lies in its questioning the habits of the architects.

Default: Analog/Digital

Architecture is an analog affair. In order for a building to be big (compared to its surroundings), it has to be built big. In order for a building to be built narrow, it has to be narrow. A “thick” wall necessarily has to be drawn and built thick. Language, on the other hand, is digital. The word “big” is smaller than the word “small.” To visualize a red interior, the architect has to use red color markers or pastels or pencils to sketch it out in red. Architects are used to analog modes of visualization and building. It is not in the DNA of architects to be colorblind.

AutoCAD is colorblind by default. It is also lineweight-blind by default. While ‘color assigned by layer’ facility was available from AutoCAD’s first version, entities having their own color irrespective of the layer on which they reside was introduced only AutoCAD Release 7 (V. 2.5). An examination of John Walker’s logs reveals the following deliberation in the list of “Questionable Items”: “Should entities have colors? Should color be associated with an entity rather than with a layer?”

The concept of *line weights* was introduced for the first time in version 2000. It throws the analog architects into a digital world where colors and line weights either do not exist or do not correspond to each other or are achieved and coordinated with great difficulty. By subverting analog thinking, AutoCAD unwittingly does a Derrida on design process.

Epilogue

Deconstructing just the interface of AutoCAD has yielded a plethora of critical insights into the software’s relationship to architectural design process. In the end, the critical value of AutoCAD depends on how the user unorthodoxly and reflectively engages the software in the design process, which sheds light on the user’s own process of design. Much more could be understood if the deeper structures and design of the software are subjected to a lengthier critical close reading.

Methodological Issues for Further Research

One of the intentions behind writing this paper was to put forth a larger critical framework to critique software (design). For further research, a number of questions could be raised based on the present paper. What are the *aporias* or irresolvable internal inconsistencies fundamental to the functioning of the software? What are the assumptions that the software makes in “constructing” the program? How does one deconstruct software? How can one deconstruct *using* the software as a critical and/or political tool? Does one look into the text of the code? Or does one look into its mere manifestation when it is “run”? Where does one intersect the hidden agendas of the software? Following Derrida’s conclusions that eliminate the notion of geometrical truth and certainty, how does one deconstruct the geometric modelers? What IS software and what are its boundaries? At what point does the “text” of the software become the “quasi-space” of its run-time self? Should the code or the runtime response be taken into account? Should one look into the institution of its making also? Should software, which is a quasi-spatial construction, be subjected to similar critique as Michael Benedikt did with the Kimball Museum?

These questions obviously go beyond the immediate goals set forth for this paper, but will serve to forward a direction of critique of software in general.