XIV del 1 al 5 de diciembre de 2008
CONVENCIÓN CIENTÍFICA
DE INGENIERÍA Y ARQUITECTURA

ANIVERSARIO 44 cujae 2008

Gulliver in the land of generative design

Charles C. Vincent

***Abstract* — The current trend in architectural design towards architectural computing has been treated both from a philosophical standing point and as an operational systems' problem, in a quest for explications which could at last break ground for a more broad development and adoption of digital design tools. But the intuitiveness that architects have put on so high a pedestal seems to be the central issue to be dealt with by both views. The very foundations upon which we prepare future professionals might change, not only in College, but also in High School. In this paper, we delve further into the discussion about the disconnect between current curricula and digital design practices and suggest new disciplinary grounds for a new architectural education.**

***Key Words* — Digital Design, Humanist Computing, Intuitiveness, Digital Education, Architectural Curricula.**

I. Introduction

The current trend in architectural design towards architectural computing has been treated both from a philosophical standing point and as an operational systems' problem, in a quest for explications which could at last break ground for a more broad development and adoption of design tools.

As Kostas Terzidis (2007) puts it, the intuitiveness that architects have put on so high a pedestal seems to be the central issue to be dealt with by both views. There seems to be no apparent shortcut toward the reconciliation between traditional practice and new media and most certainly it is not only a problem of interface design, but one of design method clarification and reinterpretation of those methods into computing systems. Furthermore, there's no doubt left as to whether computing systems can generate such new patterns as to impact our own understanding of architecture.

But even if computer algorithms can make possible the exploration of abstract alternatives to an abstract initial idea, as in Mathematica, Processing or *vvvv*, to name a few, the issue of relating abstract and geometric representations of human centered architecture lays in the hands of architects, programmers or, better yet, architect-programmers. What seems now to be the relevant change is that architectural design might escape from the traditional sequence embedded in the *need – program – design iterations – solution* timeline, substituted by a web of interactions among differing experimental paths, in which even the identification of needs is to be informed by computing.

It is interesting to note that the computational approach to architectural design has been praised for the formal fluidity of bubbles and Bezier shapes it entails and for the overcoming of functionalist and serialization typical of modern architecture. That approach betrays a high degree of *canonic fascination* with the tools of the trade and very little connection to the day to day chores of building design. On the other hand, shall our new tools and toys open up new ways of thinking and designing our built landscape?

What educational issues surface if we are to foster wider use of the existing technologies and simultaneously address the need to overtake mass construction? Is mass customization the answer for the dead end modern architecture has led us to? Can we let go the humanist approach begun in Renascence and culminated in Modernism or shall we review that approach in view of algorithmic architecture?

Let us step back in time, to 1726 when Swift's 'Travels into Several Remote Nations of the World by Lemuel Gulliver' was first published. In Swift's fierce critic of what seemed to him the most outrageous ideas, he conceived a strange machine devised to automatically write books and poetry, in much the same generative fashion that now, three centuries later, we begin to cherish.

"Every one knew how laborious the usual method is of attaining to arts and sciences; whereas by his contrivance, the most ignorant person at a reasonable charge, and with a little bodily labour, may write books in philosophy, poetry, politicks, law, mathematics and theology, without the least assistance from genius or study. He then led me to the frame, about the sides whereof all his pupils stood in ranks. It was twenty foot square, placed in the middle of the room. The superficies was composed of several bits of wood, about the bigness of a dye, but some larger than others. They were all linked together by slender wires. These bits of wood were covered on every square with paper pasted on them; and, on these papers were written all the words of their language in

their several moods, tenses, and declensions, but without any order. The professor then desired me to observe, for he was going to set his engine at work. The pupils at his command took each of them hold of an iron handle, whereof there were forty fixed round the edges of the frame; and giving them a sudden turn, the whole disposition of words was entirely changed. He then commanded six and thirty of the lads to read the several lines softly as they appeared upon the frame; and where they found three or four words together that might make part of a sentence, they dictated to the four remaining boys who were scribes. This work was repeated three or four times, and at every turn the engine was so contrived, that the words shifted into new places, as the square bits of wood moved upside down." (Jonathan Swift, Gulliver's Travels, A Voyage to Balnibarbi)[i]

What astonishing forecast did Swift show in that narrative that, in spite of the underlying incredulity and irony, still clarifies our surprise when faced to what might seem to some of us just an abandonment of all that architects and designers have cherished: creativeness and inventiveness.

Yet, we could argue that such a radical shift in paradigm occurred once when master builders left the construction ground and took seat at drafting boards. The whole body of design and construction knowledge was split into what now seem to us just specialties undertaken by more and more isolated professionals. That shift entailed new forms of representation and prediction which now each and all architects take for granted. Also, Cartesian space representation turned out to be the main instrument for professional practice, even if one can argue that it is not more than an evolution of stone carving techniques that master builders and guilds were so fond of.

Enter computing and all its unfolding, i.e. DNA coding, fractal geometry, generative computing, nonlinear dynamics, pattern generation and cellular automata, as a whole new chapter in science, and compare that to conical perspective, descriptive and analytical geometry and calculus, and an image begins to form, delineating a separation between architect and digital designer, much in the same fashion architects separated from master builders.

II. Materials and Methods

It's difficult for architects and educators whose education fostered geometry and formal drafting as key tools for architectural practice to encompass more abstract ideation tools.

Design processes have been taken quite often as *black box* processes, in spite of some theorists' arguing in favor of *glass box* methods. The black box approach is quite similar to what traditional architects and studio coaches attempt: the ideation is preceded by carefully collecting data regarding architectural needs, studying similar cases – both in search of known typologies and constructive solutions – , and then letting solutions surface, or emerge, in sort of a divinatory process. There's little, if any, tentative of making the ideation itself controllable or predictable.

The emergence of a broad and ever growing assortment of tools and associated practices, as shown in *white papers,* communications, software briefs, case studies and so forth, have been mediated by discourses in an attempt to unify views and produce an acknowledgeable set of practices. Among these emerge *glass box* methodologies, derived in great part from operational research, as old fashioned it may be, and these methodologies are understood as necessary since architectural software such as BIMs and their close relatives *Generative Doe* favor a deeper control of their inner processes. How to relate the *design process* to the *computation process* remain an open question, though.

Rivka Oxman (2005) has already touched this ground when referring to the need of explication of digital design processes. Her schematic approach for the explication of the connections among architects and differing types of digital tools can be unfolded onto more detailed diagrams describing the flux of information during architectural ideation.

Besides, one of the interesting studies towards the clarification of architects and designers' thinking is that of Paul Lasseau (1986) and the transposition of some of his *graphic thinking* into software might prove invaluable to reconcile traditional practice with parametric and generative design. As Lasseau puts it, graphics are a powerful tool both for analytic and synthetic thinking processes, which is what renders sketching and drafting the *tools of the trade* for our practice.

But until some years ago the chore of *sketching* in software like Autocad and other *geometric calculators* has proved inefficient and thus precluded most architects of beginning design tasks directly in CAD.

Some attempts to supersede those limitations were devised in a very crude fashion in software like Autocad Architecture's *Space* object and Vectorworks' *Space Planner*. The idea of representing more abstract requirements such as spaces (and not their enclosing constructions) offered architects some loose tools to start design directly in those software. However, the formal limitations imposed by their geometric engines and the impossibility of altering the logical connections between preliminary space design and formal and conceptual design, i.e. altering propagation rules, greatly limited their adoption as *real design tools*.

This seems be the case with Generative Components, Paracloud and Grasshopper. As out-of-the-box solutions, those pieces of code try with varying degrees of complexity to expose the parametric nature of architectural modeling. Even BIM software such as Revit manage to keep an open door to inner programming – both at a very shallow depth, in the case of family creation, and a deeper one, accessible only to those versed in VBA.

The first and more publicized benefits of these software have been related to the flow of information from form conception to construction planning, or fabrication.

This tack leads us back onto an often forgotten triad – *venustas, firmitas, utilitas* -, of which apparently only the first

two parts have gathered enough attention, the third one being frequently left aside: *utilitas*. As broad as we may take their meanings, these words might now suffice to encompass issues in architecture that should be tackled in software.

Although it is relatively easy to devise performative mechanisms in which a piece of code evaluates constructive geometry against, for instance, solar incidence and calculate heat gain and illumination levels and return feedback to the geometry engine, altering parameters and rerunning the process until a solution emerges, the same cannot be said of cognitive issues. Those require more abstract representations of the kind Laseau has explored extensively.

The very possibility of articulating rough graphic representations of *functional* and very abstract issues to more detailed *digital modeling* will force us to review the old question of *black box* versus *glass box* methodologies, since the parametric connections between design intent and design representation – or better yet, design prototyping –, will demand clearer and more explicit reflections about the process. [2]

This might sound canonical, certainly. However, this very canonical approach reflects our intuition towards the need to discuss and evolve our own tools and this might be taken as an human evolutionary characteristic as well.

III. Results and Discussion

The yet quite limited practice as carried on at our school has been taken in the form of free exercises proposed to a few interested students, outside the classroom. We've developed a feeling that "the general impression resulting from those experiments is that the correct approach involves teaching digital tools in a design ambience, i.e. proposing designs where the impact of digital tools is decisive in the formation and development of concepts." (2008: Nardelli, Vincent) As a result of yet unconcluded studies we try to show some of those connections, as experimented within Generative Components, Rhino+Paracloud and Rhino+Grasshopper.

At the present moment, those few students delving into the exploration of these tools face a twofold drama: on one side, the dullness of software interfaces imposes an additional layer of unnecessary complexity to otherwise simpler tasks. On the other side, their lack of *canonical* reflections on design methods render the task of representing ideation of processes a most difficult one, even when considered that those few students are some of the best in software learning. Like all new concepts, the idea of representing a process is hard to grasp at first hand. Our brief experimentation with the new tools is summarized in the following paragraphs.

In Bentley's GC, the hierarchic parametric diagrams are displayed using the graphic engine of Microstation, and its responsiveness to mouse input is limited to the arrangement of icons for clarifying the structure. One cannot alter the relations in a graphic way. On the other hand, mouse access to geometric entities in the model area is granted.

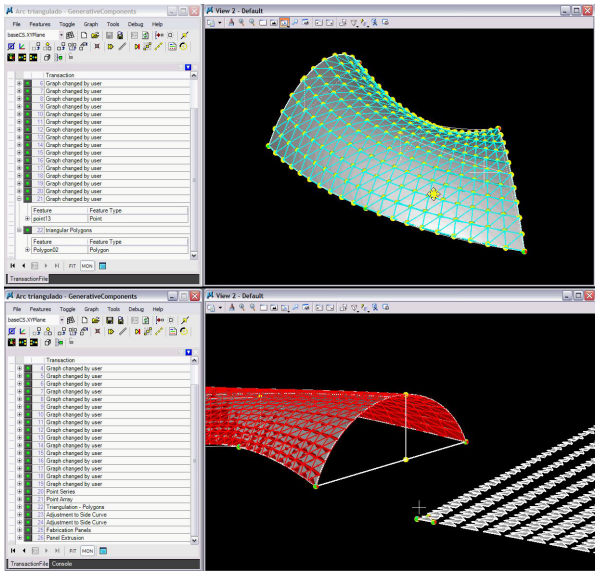

Fig. 1 - Bentley's Generative Components - first runs.

CG is heavily dependent on transactions, which might appear as a barrier to the more fluid architects' graphic processes. However, the parametric connections between more abstract graphs – such as, for instance, splines representing traffic flows, boxes representing air volume requirements –, greatly facilitate the architects' job of associating functional concepts to form generation and, once the connections are established, mouse input serves well as a mean of altering *base geometry*, with instant propagation onto more complex forms.

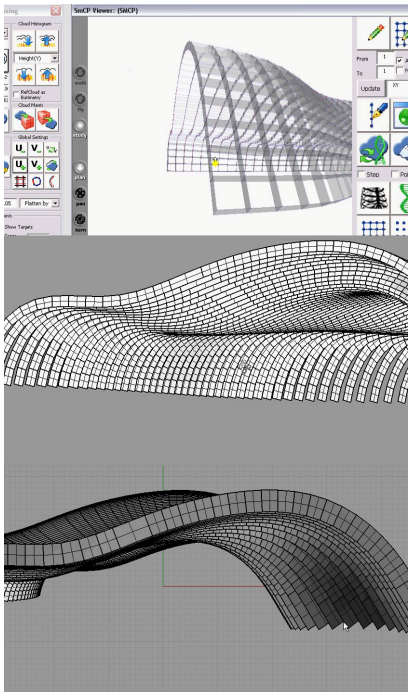

Fig. 2 - Paracloud : student first tests.

Paracloud, as far as we could see, provides quite efficient methods for analysis and transformations – by means of matrices – applied to formal geometry. But its interface is dull

and not even close to 'fluid'. We would say it still needs some degree of developing, particularly as its graphics engine – VRML Cortona –, is crude and behaves unpredictably in response to mouse action.

The most recent addition to the generative branch is Grasshopper, for McNeel's Rhinoceros. Again, our experience with it has been rather tentative, but we could advance some first impressions of its behavior. Geometry is created, in a similar fashion to GC, by *declarations*. The difference this time is that, unlike GC, both the declarations and the logical structure of dependencies are unified into a single area. The closest interface to Grasshopper would be *vvvv*, a graphic programming engine hardly useful to architects. As such, our experience with it has been pretty straightforward, since the whole logic structure is clearly displayed and fully manageable.

IV. Conclusion

What in BIM software are catalogs of building components, parametric or not, might find its counterpart in *conceptual* objects for ready use. Programmable space connectors depicting, for instance, the *ideas* of door, window – and not the *objects* doors and *windows* –, flow splines, depicting the flow of people inside a building, through promenades and concourses, and so on.[3] The average architect would resort to a rather comprehensive conceptual vocabulary and its connects. Preliminary design could then be just a matter of connecting and establishing quantities for the *objects*. After sketching with such objects, connections from roughs to forms is to follow and then from form to constructs. As of our experimentation, the representation of these concepts is made with geometry, to which constraints are applied and from which other constraints are linked to more evolved constructive geometry. The real intent is not embedded into the geometry, depending largely on the user choice.

It seems that software developers are yet to come close to more intuitive interfaces, which could seduce the average architect into the digital realm. Such phenomenon is to be quite similar to what once happened to web design: in the first years, web sites revealed more of their internal html logic; after those beginnings, designed interface layers begun being superimposed on top of the harsher structure, greatly facilitating widespread use. Finally, we observe today an increasing number of web design tools which further cover the html and xml logic, broadening design possibilities and domesticating programming languages.

What did Swift skip in his skeptic description of *Academy of Lagado*'s random writing machine? A shape grammar? Or perhaps some semantic system? Would we survive his keen scrutiny?

Acknowledgment and Notes

[1] P.S. Curiously enough, this is the same parallel William J. Mitchell traced in the last paragraph of his *The Logic of Architecture – Design, Computation and Cognition,* which I rather unconsciously retrieved from some deeper cerebral registry.

[2] It is necessary to stress the difference between constructive functionality and *human use* functionality. When referring to function in architecture we intend it to be taken as related to human use and appreciation of buildings.

[3] It is arguable that these processes are still representational ones, the main difference being that the represented subject is now the ideation itself, not the buildings. This might pose some problems, since much of academia is trained to deal with the objects, and the critics made upon students' production is a critic of the object, with little emphasis on the understanding of creative processes.

References

[1] Terzids, K.: 2007, Digital Design: Ideological Gap or Paradigm Shift?, in *La Comunicación en la Comunidad Visual*, Proceedings of the XI SIGraDi Congress, Universidad La Salle, pp. 220–224.

[2] Oxman, R.: 2006, Theory and Design in the First Digital Age, in *Design Studies*, Vol. 27, pp. 229–247, Elsevier.

[3] Nardelli, E.S., Vincent, C.C.: 2008, Diagnosis and Strategies for a Digital Design Studio, in *Architecture in 'computro' – Integrating Methods and Techniques,* proceeding of eCAADe 2008, Antwerp, pp.177-182.

[4] Laseau, P.: 1975, *Graphic Problem Solving for Architects and Builders*, Boston: Cahners Books International.

Charles C Vincent is an architect and lectures at the School of Architecture and Urban Planning (FAU), Universidade Presbiteriana Mackenzie, São Paulo, Brazil. He holds a doctoral degree from Universidade de São Paulo, where he also graduated. Presently he is Executive Editor of *Cadernos de Pós-Graduação in Architecture and Urbanism*, and researcher with the *Teoria e Projeto na Era Digital* group, at Universidade Mackenzie.