BETWEEN ACCIDENT AND CONTROL: CONTRASTING TRADITIONS OF COMPUTATIONAL DESIGN

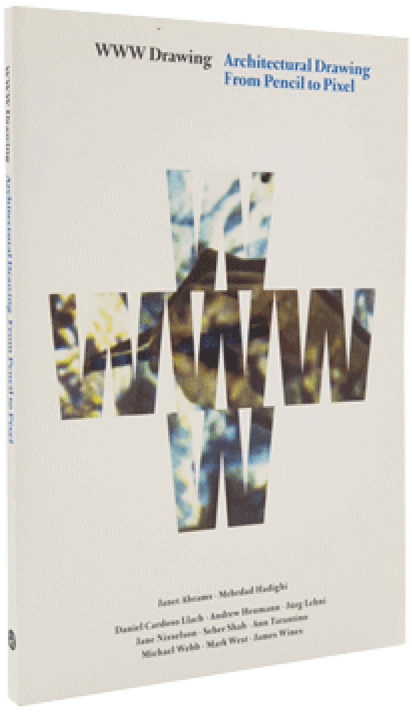
Posted on October 16, 2020 by martabuges

Categories: Daniel Cardoso-Llach, Essay, expanding design practices, No Density, Technology and fabrication, WWW Drawing

Tags: Algorithms, Analysis, Architecture, Automation, Challenge, Comfort, Communication, Computation, Design process, Design thinking, Digital era, Drawing tool, Engineering, Essay, History, Mapping, New technologies, Research, Simulation, Smart tool, Technology, Transformation

Authorship: This essay by Daniel Cardoso-Llach, is an excerpt of WWW Drawing by Janet Abrams and Mehrdad Hadighi, published by Actar Publishers and Pennsylvania State University's Department of Architecture, Stuckeman School of Architecture and Landscape Architecture. Learn more:

ISSN : 2575-5374


WWW Drawing
Architectural Drawing From Pencil to Pixel
WWW
Janet Abrams · Mehrdad Hadighi
Daniel Cardoso Llach · Andrew Heumann · Jürg Lehni
Jane Nisselson · Seher Shah · Ann Tarantino
Michael Webb · Mark West · James Wines

ISSN : 2575-5374

Since architects began to flirt with computers in the 1960s, debates about the role of computation in architecture have often been framed antagonistically – as arenas for technophobes and technophiles to clash, each staking a claim on the unique value, or promise, of their respective practices. This dichotomy is tempting. On the one hand, images of automated design systems – offering creative freedom, managerial efficiency, or 'personalized' design solutions – abound in architecture's six-decades romance with computation. Seductive and often reductive, these images outlined the contours of a computationally augmented practice of architecture, and captured the imagination of many architects in academia and industry, effectively ushering an entire academic sub-field.

ISSN : 2575-5374

Fig. 1: A reconstruction of Ivan Sutherland's 1964 "Sketchpad" system, the first interactive graphics system, sits alongside Blooms Day, a 1969 generative painting by George Stiny, in Designing the Computational Image, Imagining Computational Design, an exhibition curated by Daniel Cardoso Llach in 2017 at the Miller Gallery, Pittsburgh.

Computers were greeted more cautiously, on the other hand, by those architects who saw them either as means of producing ineffective simulacra, or as a dangerously transformative force in architecture. For some in this group, computer screens simply failed to approximate the plasticity of sketches drawn by hand, or the tactility of a physical model; for others – aware of the social and organizational tensions introduced by technologies – computers conjured (not entirely unjustified) pre-industrial fears of automation, de-skilling and alienation.

A sort of bargain was thus struck across these seemingly distant intellectual territories: the idea that the computer is 'just another tool' that architects can utilize in their design process – another paintbrush or easel in the architect's atelier. Frequently deployed in studio reviews, 'think pieces' and course syllabi, this concept has become part of many architects' conventional wisdom, configuring a comfortable middle ground where computational ideas and techniques can co-exist with (albeit at a safe distance from) architecture's hard core.

My argument here is that, rather than clarifying, the 'just another tool' discourse obfuscates the truly important questions concerning computation in architecture. By casting hand-drawings and computer-generated images as symmetrical, it renders the very specificity of computation – a domain of analysis with historical and cultural depth – invisible. If we are to understand and address the contemporary entanglement of computation with architecture's long standing intellectual traditions and embodied practices, we must make computation visible and challenge the dichotomy that forces us to see it as either a threat to architecture's core, or as the protagonist of all-encompassing historical turn.

One of the numerous consequences of the 'just another tool' blind spot is the encouragement of unproductively nostalgic 'post-digital' attitudes. More importantly, by framing computers as 'just another tool' we make the infrastructural scale of computational design technologies invisible, and thus postpone, rather than address, a crucial debate.

In this essay I want to focus on specificity, difference, and (to borrow a term from aeronautics) the dissymmetries between hand and computer drawing. The crucial difference between computer-generated images and hand drawings is that the former have structure.

ISSN : 2575-5374

We may visualize this through an architectural metaphor. In a building, the structure is the rigid skeleton – usually made of wood, steel, or concrete – that makes it stand. Similarly, a computer drawing is unthinkable without an underlying structure. This structure is made not of wood or steel but of symbols; it is computable and numerical, and is encoded in the non-pictorial languages used by computers and software: it is made of code. Similar to a building whose structure is concealed beneath cladding and paint, the structure of a computer drawing is hidden from view, and is fundamentally different from image itself.

This becomes obvious when we consider how different the symbols are from the image we see glowing on the screen. This decoupling of the image and its structure is not an opinion, nor a theoretical construction, nor a value judgment. It is the fundamental fact of computer graphics – and the crucial, irreducible difference between hand and computer 'drawings'. Compared to computer-generated images, hand drawings have no structure. As CAD pioneer Ivan Sutherland notably put it, "they are only dirty marks on paper." Acknowledging this distinction helps dispel the deceptive symmetry of the 'just another tool' discourse, and allows us to examine the issues at stake in more detail, and makes the instruments involved in computational design visible. I am speaking, of course, of software and, more specifically, of the software interfaces that strongly condition architectural labors today by structuring the experience of computation for users. Software interfaces shape the way computation's own materialities – electrons, switches, logic gates, machine code, data structures, and software functions, roughly in that order of abstraction – intertwine with the materialities of design and thus are important to our analysis (Fig 2).

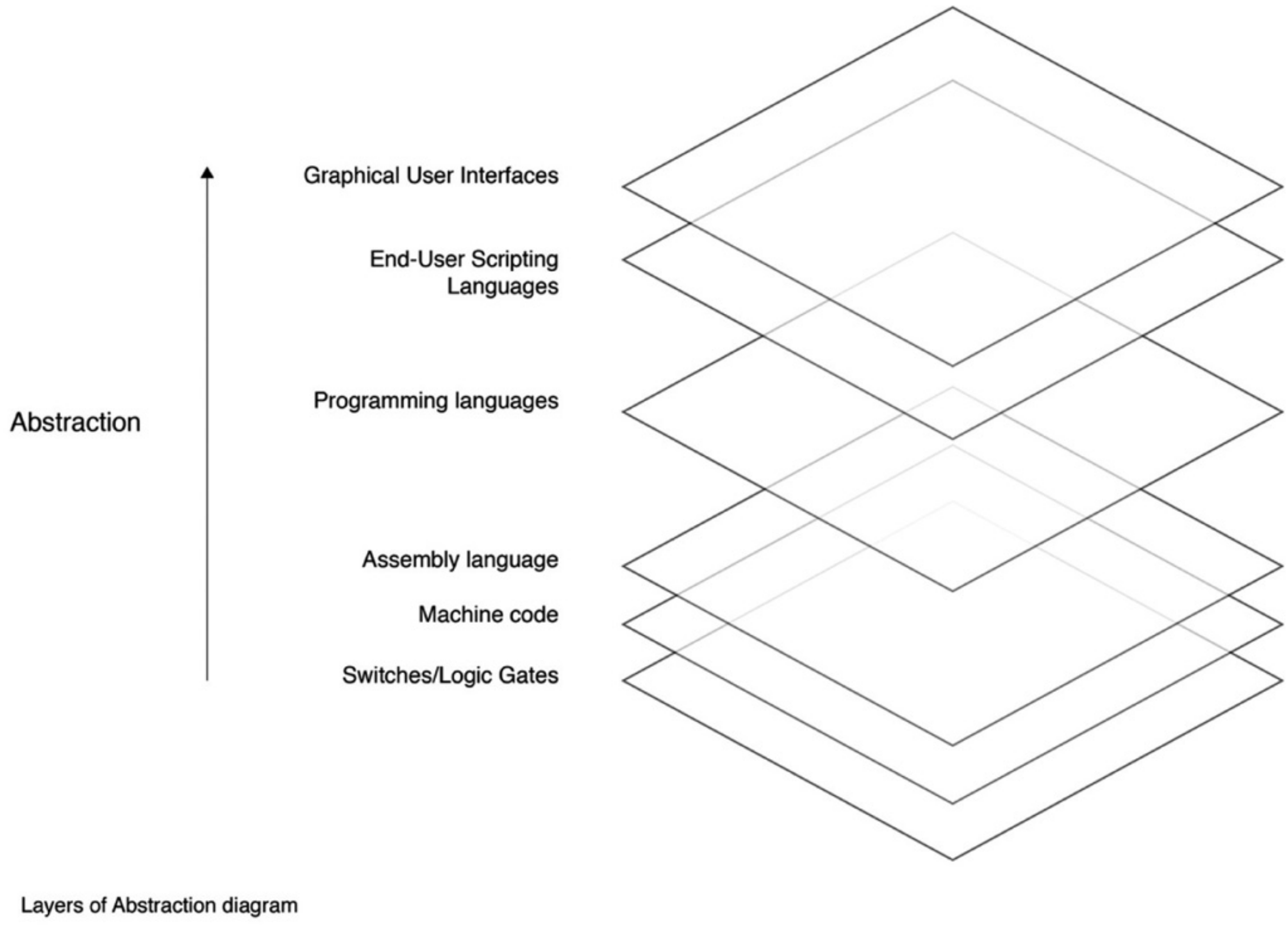

Graphical User Interfaces
End-User Scripting Languages
Programming languages
Abstraction
Assembly language
Machine code
Switches/Logic Gates
Layers of Abstraction diagram

Fig. 2: Diagram representing the varying levels of abstraction of computational systems. The higher levels of abstraction, such as Graphical User Interfaces, are typically both human-readable and hardware-agnostic. Image: Daniel Cardoso Llach.

Rather than as pawns in a tired culture-war, we ought to see software and software interfaces as active participants in the worlds of architecture – in fact, as the very infrastructure of many

ISSN : 2575-5374

architectural activities – and as historically and culturally situated theories of design. In an attempt to shift the focus of the discussion, in this short essay I will sketch some underpinnings of software qua theory of design in architecture by discussing two distinct intellectual traditions of computational design.

The Algorithmic Aesthetics Tradition

The first I shall call the 'Algorithmic Aesthetics' tradition. Influenced by Noam Chomsky's theory of Generative Grammars in linguistics, and by George Birkhoff's numerical theory of aesthetics, German Philosopher Max Bense first formulated the concept of "generative aesthetics" in an influential 1965 manifesto. Bense, who was based in the University of Stuttgart, Germany, saw computers as vehicles of aesthetic investigation. His teachings helped spur a generation of pioneers – including Frieder Nake, Georg Nees and Vera Molnar – who used early pen-plotters to produce some of the earliest examples of computer-generated art. Often using pseudo-random numbers to determine the placement of visual elements in their compositions, the work of these early computer artists hinges on a delicate balance between regularity and disorder (Fig 3).

ISSN : 2575-5374

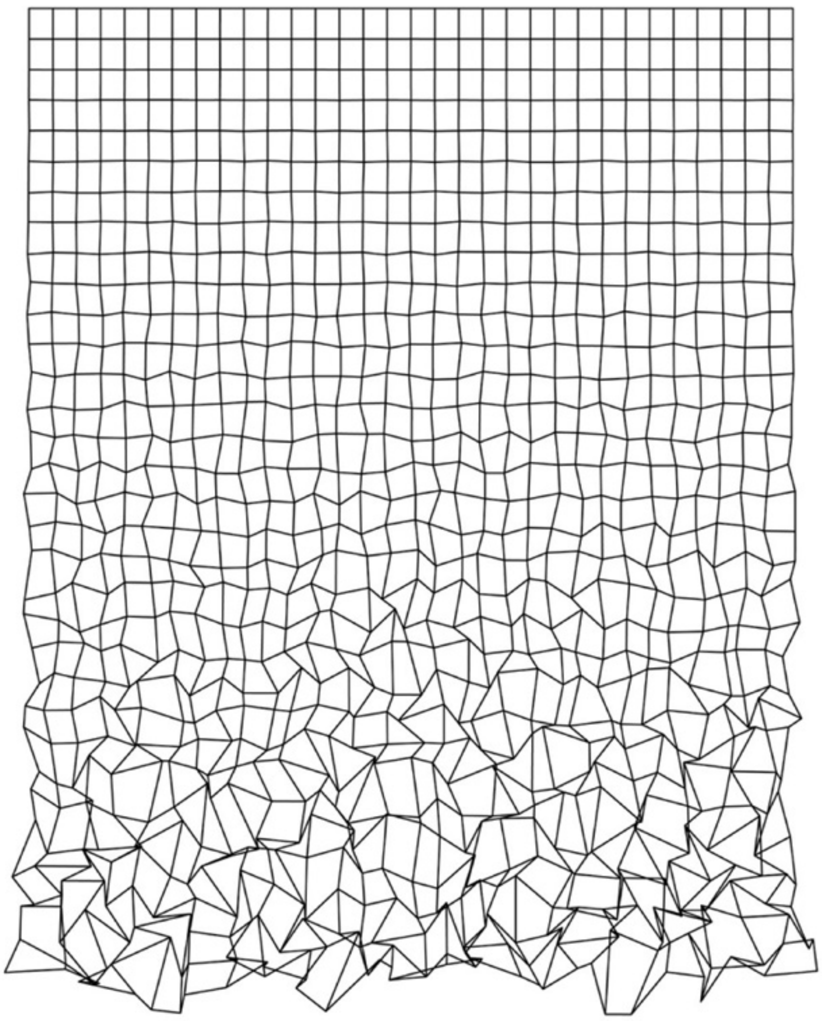


Fig. 3: In a course taught by the author at Penn State University, students reconstructed early works of software art. This image is a 2013 reconstruction in the Processing language of Aesthetic Unrest: Dispersion of Squares (1968) by computer art pioneer Georg Nees. Reconstruction by Seoug Oh.

Their work was received with skepticism by the art world. Two factors may help explain this resistance. On the one hand, its procedural nature challenged conventional understandings of artistry, skill, and authorship. On the other, an influential segment of the artistic establishment of

ISSN : 2575-5374

Post-War Europe saw these artworks as byproducts of the war machine: outputs of chiefly militaristic – and often US – technologies, or seductive honey traps set-up by ruthless forces of production. Despite the skepticism it encountered initially (some of which persists today), the work of these artists was echoed by other artists and architects; it had a considerable impact on design culture and education (for example through figures like architect and computer graphics pioneer John Lansdown in the UK) and is widely recognized as pioneering today.

In the United States, the Algorithmic Aesthetics tradition in visual design can be traced the most clearly to the work by George Stiny and James Gips, whose seminal 1971 paper on Shape Grammars spoke of the "generative specification of painting and sculpture." Like Bense's information aesthetics, Stiny's and Gips mathematical definition of design as a rule-based visual and perceptual calculation invoked Chomsky's and Birkhoff's ideas, and resonated with other contemporary intersections of mathematics and art. Because of their capacity to act as descriptive, analytic, and generative visual systems, Shape Grammars triggered a school of design practice and scholarly thought that continues to evolve today.

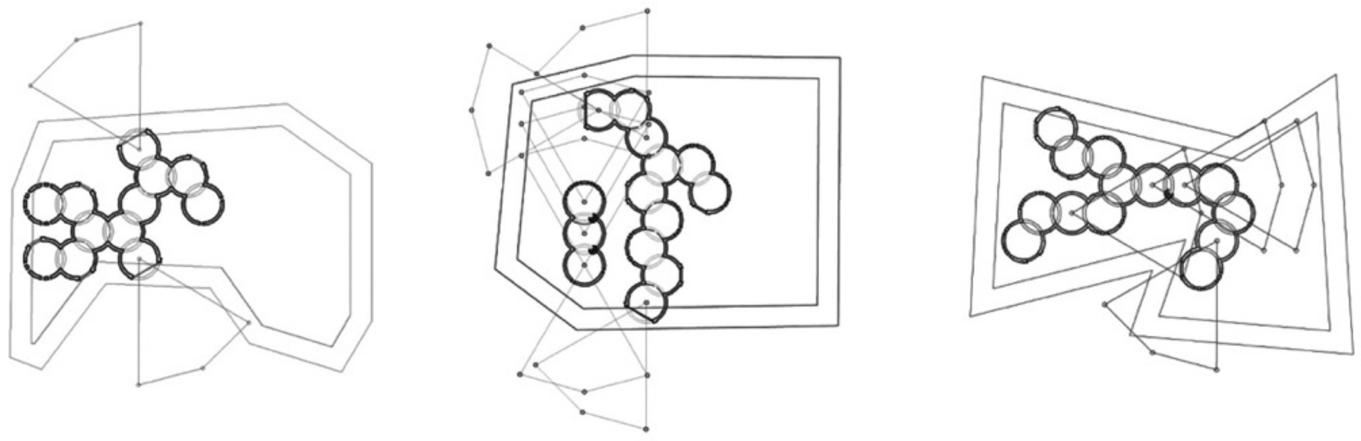


Fig. 4: The Melnikov Grammar is an experimental computer program that implements a shape grammar to generate architectural plans following stylistic traits of Russian architect Konstantin Melnikov procedurally and non-deterministically. Image and software credit: Daniel Cardoso Llach.

Among the myriad cultures of computational design that have developed over the last several decades, many evoke the Algorithmic Aesthetics tradition. These cultures have often evolved in

ISSN : 2575-5374

conjunction with open source software development languages such as Processing and Open Frameworks, and are as diverse intellectually as they are technologically. Their members' complicated disciplinary identities are often amalgams of graphic, interaction, and architectural design, as well as advertising and new media art. The lively "New Aesthetic" conversation, which debated the cultural significance of computing in the arts the late 2000s and early 2010s, reflects this sensibility. Architectural expressions of the Algorithmic Aesthetics tradition may seem hard to find at first, but they abound, especially among architectural theorists and researchers. Aside from the significant body of descriptive and analytical work in architectural studies related to Shape Grammars, researchers since the 1960s have used computation to explore 'design spaces' defined through combinatorics and enumeration; to produce non-deterministic formal or material effects; and to elicit unconventional configurational logics. (Figs. 4 and 5) Cambridge architect and researcher Phillip Steadman offers a crisp illustration of this sensibility when he defines architecture as "the science of possible forms."

ISSN : 2575-5374

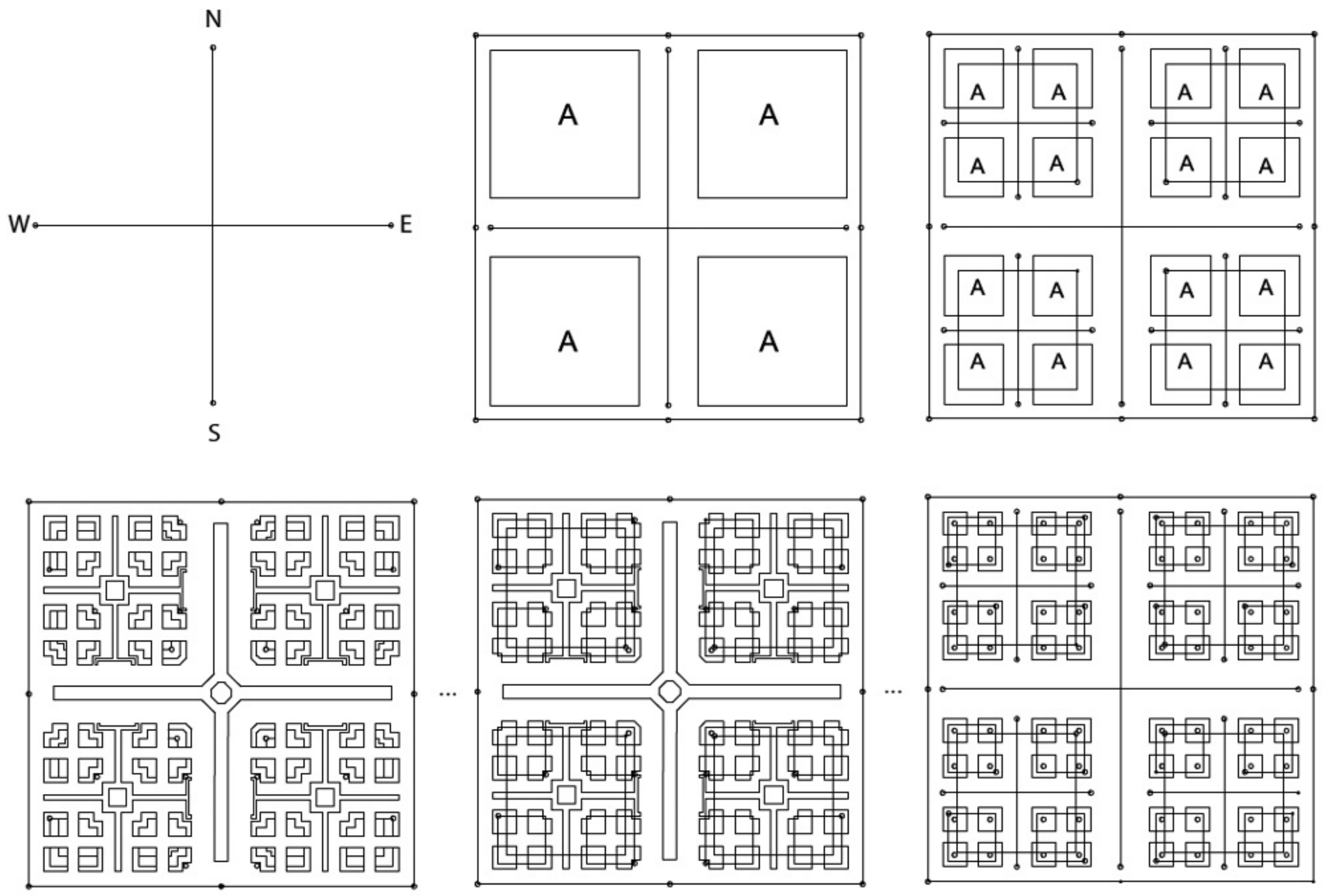


Fig. 5: Mughal garden designs generated semi-automatically using GRAPE, a computer program for implementing parametric shape rules. Image Credit: Nirvik Saha. See Economou, Athanassios, and Thomas Grasl, "Paperless Grammars," in Computational Studies on Cultural Variation and Heredity, edited by Ji-Hyun Lee, 139–60. Singapore: Springer Singapore, 2018. https://doi.org/10.1007/978-981-10-8189-7_12.

The detailed explication of these subfields could be the subject of an entire dissertation. What is key for this analysis is that for artists, architects, designers and researchers working within the Algorithmic Aesthetics tradition, computation appears as a new, open-ended medium with its own expressive, material and technical capacities and constraints. These constraints manifest through specific computational approaches to design including formal systems, stochastic methods, and statistical methods. In their work, computation does not appear as a surrogate for drawing or other forms of design, but as a new actor with entirely different dramatic potential.

Architect and artist James Wines has described hand drawing as “the fertile territory of subliminal accident.” We may use those same words to describe those working within the Algorithmic Aesthetics tradition, for whom design is about using computation to orchestrate the conditions for the right accidents to happen.

The Algorithmic Tectonics Tradition

A second tradition of computational design engages differently – perhaps antagonistically – with the creative process. Instead of serendipity and accident, its proponents seek the safety of control. This tradition, which I shall call ‘Algorithmic Tectonics,’ is steeped in technique. The root tekton (carpenter in Greek) invokes its distinguishing trait: the understanding of computer-generated images as structured and engineered artifacts. This sensibility was central to the development of the first interactive design systems. In contemporary architectural cultures, this tradition is discernible in the ambition to use computers to achieve increased managerial efficiency and control.

The early development of Computer-Aided Design systems in the 1950s and 60s offers clues about the central motifs of the Algorithmic Tectonics tradition. As I discuss in detail elsewhere, CAD systems originated in university laboratories from an engineering impulse to increase speed and efficiency in the design and manufacture of aircraft parts. An early ambition of CAD researchers was to develop symbolic languages capable of representing any design problem, and to re-imagine design itself in technological terms. Here, drawings were no longer drawn, but built. Accordingly, the first CAD tools sought not merely to replicate traditional drawing methods, but rather to explore the specific capacities of the new platform. CAD pioneers such as Steven A. Coons and Douglas Ross, and their students, re-imagined the very concept of design in computational terms as an iterative process of representation, analysis, and materialization.

The ‘structure’ of computer-generated images was central to their vision. Early CAD researchers understood that computational descriptions were less like drawings and more like databases. A geometric model of a house, for example, could be enriched with information about materials, prices, structural calculations, and other attributes. The data structures that encoded geometric information could be manipulated computationally, thus making drawings responsive to geometric and mathematical constraints. This new view of drawings as engineered artifacts allowed CAD theorists and advocates to claim that computer-generated images bore structural (not just pictorial)

ISSN : 2575-5374

resemblance with the artifacts they described. It set in motion a technological imaginary of design and creativity that increasingly dominates present-day discussions about architectural production. (Fig. 6)

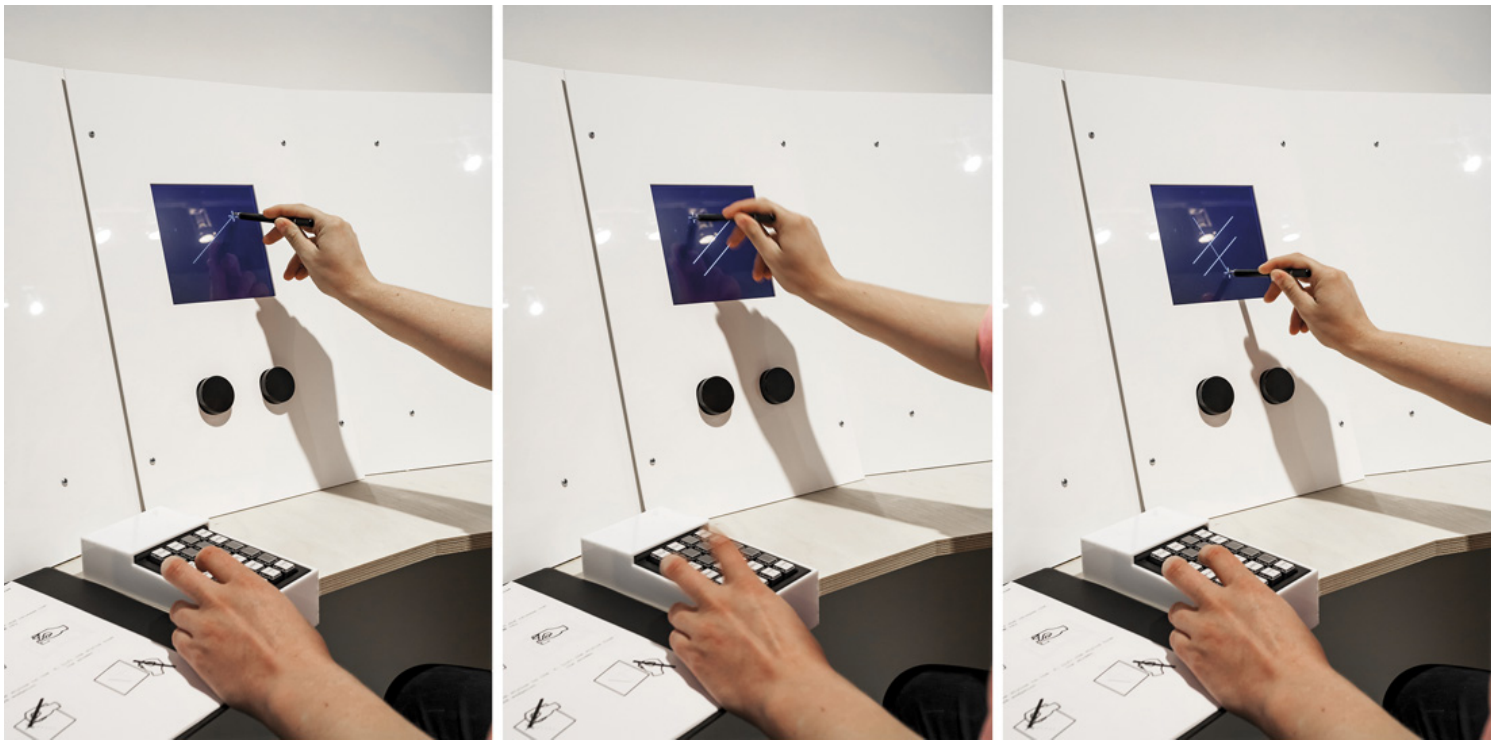


Fig. 6: Ivan Sutherland's Sketchpad is widely recognized as the first interactive graphics sofware, and as a template for CAD systems. The four drawings below were generated using a reconstruction of Sketchpad, developed by Daniel Cardoso Llach and Scott Donaldson in 2017.

This engineering sensibility toward design representations, and its commitment to managerial control and efficiency, is the essence of the Algorithmic Tectonics tradition. Through multiple cultural and historical channels – not least CAD software itself – this ambition has permeated other design fields. Architects working on 'parametric design,' for example, take advantage of the structured nature of computer-generated models to stage design processes by modeling geometric and mathematical constraints. Architects working on 'Building Information Modeling' (BIM) organize their practice – along with other professions and trades – around an interactive computer simulation so as to reduce conflicts and improve communication. (Fig 7) These forms of design

ISSN : 2575-5374

production have gained a footing in architectural practice and education, sometimes with the aura of an inevitable 'wave'.

ISSN : 2575-5374

ISSN : 2575-5374

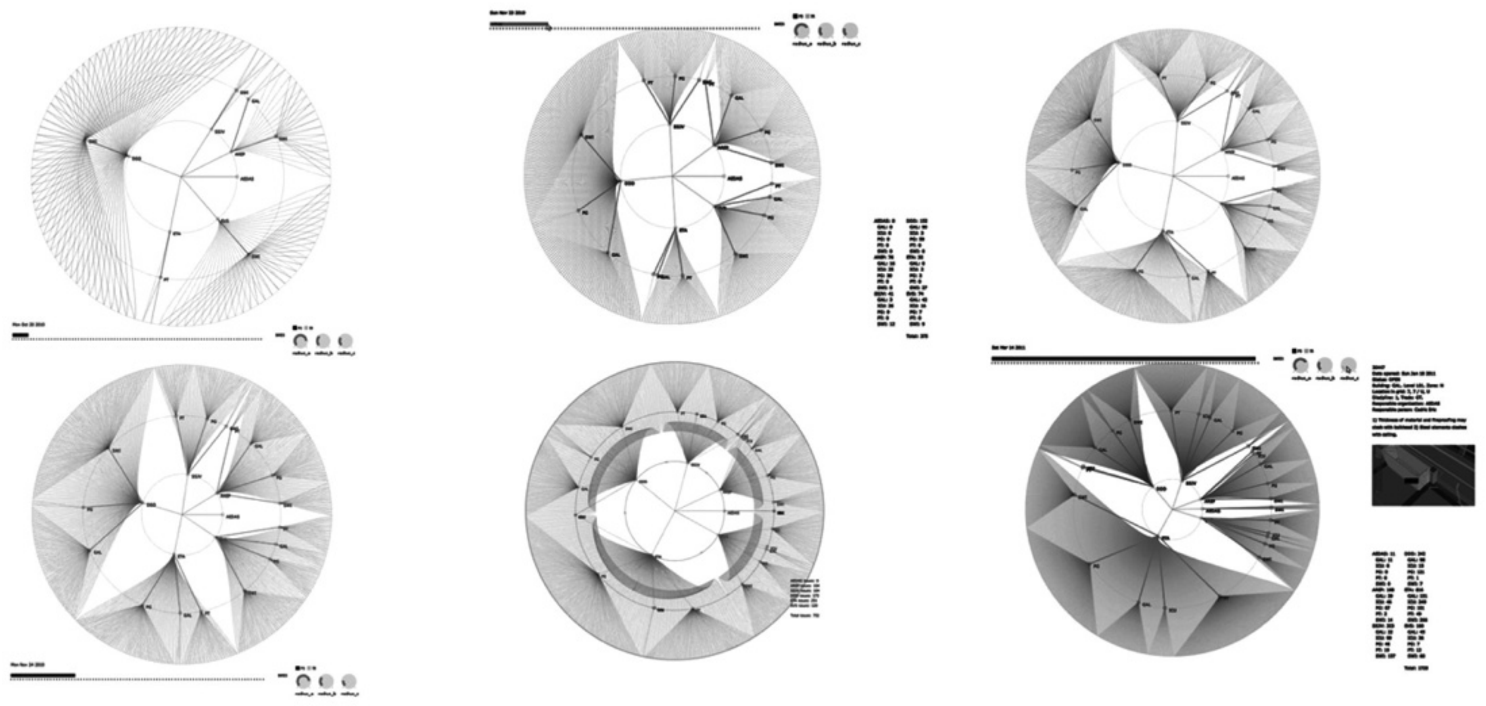


Fig. 7: These data visualizations represent different stages during the development of a large architectural project. Each line represents a design "conflict" reported during coordination. Software and image credit: Daniel Cardoso Llach, 2011.

Conclusion

While in fact intertwined, the two intellectual traditions sketched in this short essay are expressions of distinct political histories, disciplinary identities, and aesthetic sensibilities. They manifest distinct (and divergent) ideas about design and representation. They outline specific ways in which computers are not 'just another tool' but rather vehicles of theoretical, aesthetic, and practical commitments in design. To call what designers do with computers 'drawing' risks confusing expectations and limiting possibilities.

We need a new vocabulary to do justice to the unruly landscapes of contemporary computational design practices. The two sketches in this essay attempt to enrich that vocabulary.

Where the Algorithmic Aesthetics tradition is concerned with accident, unpredictability, and

ISSN : 2575-5374

authorial detachment, the Algorithmic Tectonics tradition is concerned with control, descriptive accuracy, and accountability. Where Algorithmic Aesthetics engages with open-endedness and serendipity, Algorithmic Tectonics optimizes and gives voice to our desire for certainty. Where Algorithmic Aesthetics thematizes form, Algorithmic Tectonics thematizes information.

With this enriched vocabulary as a background, we may consider the pedagogical and practical demands of a forward-looking and computationally-literate approach to architecture in a new light.

ISSN : 2575-5374

ISSN : 2575-5374