Software Comes to Matter: Toward a Material History of Computational Design

Citation: Llach, Daniel Cardoso. "Software Comes to Matter: Toward a Material History of Computational Design." Design Issues 31, no. 3 (July 2015): 41–54. © 2015 Massachusetts Institute of Technology

As Published: http://dx.doi.org/10.1162/DESI_a_00337

Publisher: MIT Press

Persistent URL: http://hdl.handle.net/1721.1/100540

Version: Final published version: final published article, as it appeared in a journal, conference proceedings, or other formally published context

Software Comes to Matter: Toward a Material History of Computational Design

Daniel Cardoso Llach

Introduction

A metaphor of weightlessness and immateriality dominates computational discourses about design. Digital information, it is often assumed, travels seamlessly through invisible networks in its disembodied binary form—existing merely as a symbolic entity. Despite recent appeals to design's materiality, particularly in discourses about digital fabrication in architecture, material formations are generally considered an *effect* of these ethereal transactions. Thus, the materiality of digital information, its (often messy) substrates—such as wires, voltages, disks, and drives, as well as the socio-technical processes involved in their definition and production—are black-boxed: hidden from view. This article explores the intellectual and material history of numerically controlled machines, and of the software that drove them, and shows that a new theoretical understanding of materials and geometry as computable, linked to the emergence of software and numerically controlled machines, emerged from the Cold War era entanglement of military, industrial, and academic interests. I show how in their quest to automate machine tools, the first numerical control researchers at the Massachusetts Institute of Technology (MIT) codified the cognitive and bodily roles of machine tool operators, as well as the properties of materials and machines, thus uncovering new questions of data storage, management, and exchange. Confronting these questions, numerical control researchers developed new languages for geometric and material inscription—*software*—that were crucially informed by the physical constraints imposed by available storage media, such as punched paper tape. From this negotiation between symbolic abstractions and material systems, new programming techniques and, crucially, the first theory of computer-aided design emerged.[1] Thus, software started to become both a vehicle for and an expression of a technical and conceptual reconfiguration of design, linked to the manipulation of materials, engineering efficiency, and militaristic control. I intend to show that software, understood as an organized set of declarative statements with both semantic and operational values, can itself be seen as a design *theory* encoding this reconfiguration.

1 The problem of the materiality of digital information is explored in Jean-François Blanchette, "A Material History of Bits," *Journal of the American Society for Information Science and Technology* 62, no. 6 (June 1, 2011): 1042–57.

doi:10.1162/DESI_a_00337

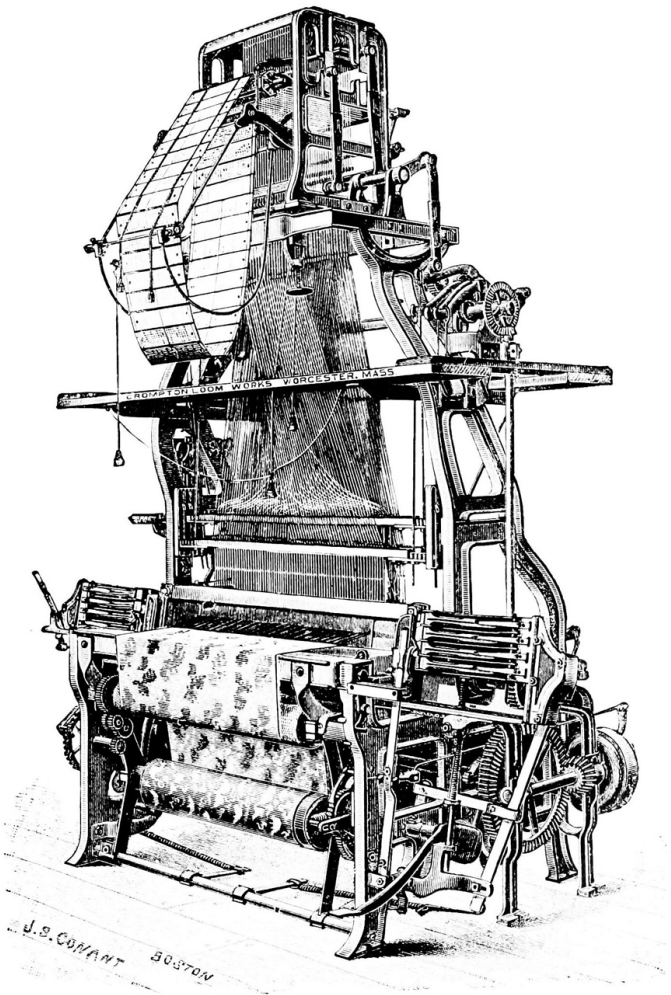

Figure 1
The Jacquard Loom. Rights: *This image is in the public domain.* From: *The Popular Science Monthly* (New York: Popular Science Pub. Co., etc., 1891), http://archive.org/details/popularsciencemo39newy.

Within this context, software's cultural significance for design is to embody a new kind of intermediary space between the messy worlds of materials and machines (metal sheets, dies, spindles, tolerances, speeds), and the clean, symbolic worlds of mathematically definable geometry. Being able to construct within the constraints of this intermediary space was the skill developed by a new social actor: the software engineer—a *craftsman* of abstraction. In the convergence of materials, bodies, machines, and geometry in the abstract worlds of computation, software literally *comes to matter.*

Software as Design *Theory:* The Origins of Numerical Control

Let's briefly consider what conventional histories of computing consider a key predecessor of software and numerical control. Long before the development of numerically controlled machinery, eighteenth century French engineer Joseph Marie Jacquard created a programmable loom controlled by sequences of punched boards (See Figure 1). The machine was able to produce complex fabrics by conditionally threading the pattern depending on whether each

Figure 2
Punched Paper Tape. Rights: *MIT Libraries, Institute Archives and Special Collections,* Cambridge, Massachusetts, Douglas T. Ross papers, MC 414, box 213. Photograph by the author. Fair Use.

of a series of needles encountered a hole (the needle goes in) or not (the needle doesn't go in). The fabric resulting from this binary conversation was as complex as the patterns inscribed in the cards. The material and physical limitations of both processor (loom) and sequential access storage media (punched cards) delineated the machine's "making" space.

The first numerically controlled milling machine—demonstrated at MIT in 1952—operated under the same principle; the programs driving the machine were stored in punched paper tape. In both the loom and the milling machine, materiality and physical constraints in the storage media determined both the kind of information stored and the range of material actions they were able to prescribe (see Figure 2).

The lively relationship between the symbolic system and its material substrate is best understood in contrast with the cognitive, manual, and embodied practices such technologies were meant to replace. Before machine tool automation, producing complex parts (e.g., airframe shapes or rotary wings) required that machine tool operators mark regularly spaced holes along a desired tool path on a two-dimensional surface (typically a sheet of metal) and then manually guide the machine to cut the part. To accurately mark the reference points, an operator had to visually obtain the numerical value of the X and Y coordinates of each point from a drawing and then calibrate the machine to the drawing's reference origin point.

By manually turning cranks, the operator could iteratively place the machine's tool head at each point, marking the points as references on the metal sheet before cutting. The process was cumbersome, and the need for repeating it led operators to use traces—templates of the desired contour of the part—to "codify" the machine's movements. Then, during production, an operator would follow the trace with a mechanical stylus (a "follower") to

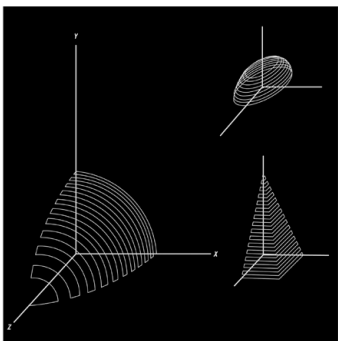

Figure 3
Author's reconstruction of an original drawing produced by the Whirlwind Computer. *Author owns rights to this image.*

drive the machine through the metal sheet. Remarkably, after decades of refining this analog process, machine tool operators in the United States had achieved precision levels within fractions of a thousandth of an inch.[2] The engineers at the Servomechanisms Laboratory at MIT sought to replace these traces and trace operators with machine-readable numerical data. The scope and intent of these efforts are concisely illustrated in Servo Lab director Gordon S. Brown's notes, scribbled on yellow paper at the onset of the project:

> The objective of the present investigation is the design of a milling machine capable of producing specific curved or irregularly contoured machined surfaces automatically without the use of models, contour cams, or other manufactured reference surfaces. In lieu of such fabricated reference surfaces, it is desired that numerical data representing the desired surface in terms of the machine coordinates will be used to guide the machine.[3]

Their course of action involved the modification of an existing milling machine so that its three basic axial motions could be controlled automatically through servo-mechanisms—a technology the eponymous laboratory had mastered during its wartime effort on gunfire control applications. They developed a system of symbols that did not simply capture the trace operator's work, but re-interpreted it.

While in the earlier, trace-controlled manufacturing processes, operators specified a series of points along the cut path to guide the machine, in the new process, the engineers—James O. McDonough and William M. Pease—developed a new "incremental-coordinate continuous-path system" describing the machine tool path as a sequence of straight segments in three-dimensional space. With this new approach, a straight cut of any length could be described by a concise dataset: two sets of spatial coordinates, direction, tool geometry, and spindle speed for each of the machine's three axial motors. In practice, the change meant that the amount of information specifying a shape was proportional to the shape's geometric complexity, and not to its size (see Figure 3). The system enabled by the new notational unit (called "command") and algorithm reinterpreted the trace operator's bodily and cognitive roles both through the incremental coordinate system and by encoding geometric, material, and mechanical constraints in a concise, mathematically precise and machine-readable notation. A small digital processor would then translate the paper tape instructions into analog signals to control the machine.[4]

This new notation was neither immaterial, nor abstract—it was shaped by material constraints in two important and distinct ways. First, tool size, material, tolerance, and spindle speed, among other parameters, had to be taken into account, therefore

2 J. Francis Reintjes, *Numerical Control: Making a New Technology* (New York: Oxford University Press, 1991), 142.

3 Servomechanisms Laboratory, "Parsons Milling Machine: Tentative Proposal," September 16, 1949, AC 151, Series II, Box 27, MIT Archives.

4 The engineers labeled this component a "director": a small, custom-designed, special-purpose digital processor that accepted machining instructions stored on paper tape in numerical form and provided analog signals to drive the machine tool. (Reintjes, *Numerical Control*, 48.) However, the incremental approach developed by the MIT researchers has been criticized as over-complicated by critics like David Noble. See David F. Noble, *Forces of Production: A Social History of Industrial Automation* (New York: Oxford University Press, 1986).

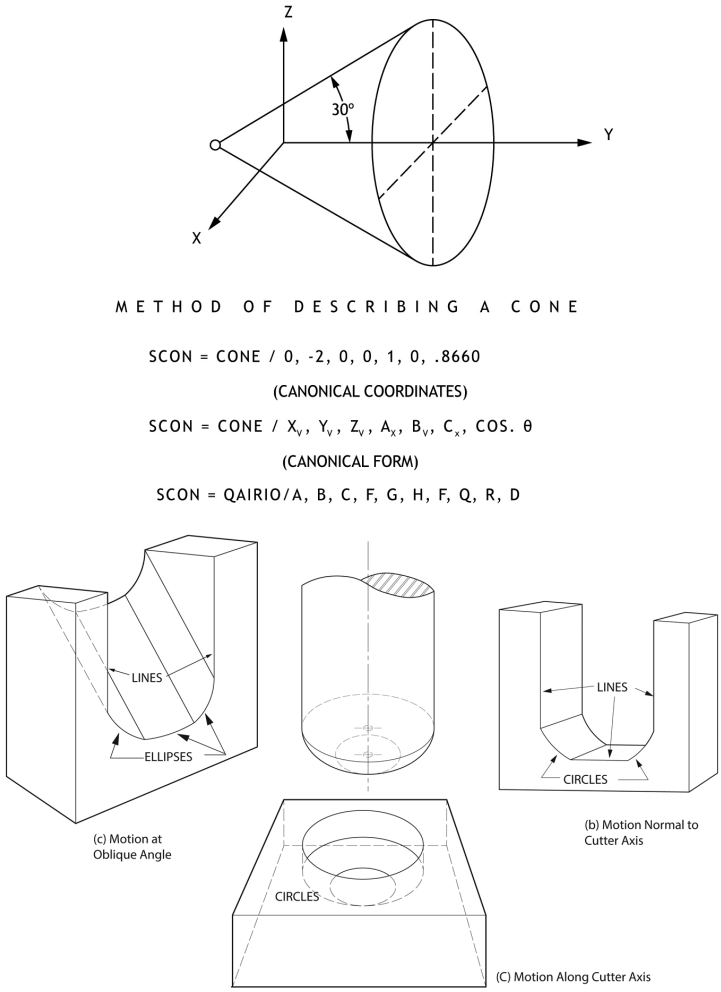

Figure 4
Author's reconstruction of an original image showing the mathematical notation of three-dimensional forms. *Author owns rights to this image.*

prompting the development of further abstractions and programming techniques. Second, the limitations in the information-storage medium—particularly tape width—demanded an efficient descriptive protocol.[5] Economy of information was crucial because each command had to fit into the width of the paper tape. Geometric, material, and machine constraints were realized as a unitary symbolic description—a "command."

As suggested, the cultural significance of numerical control for design is its illustration of an inchoate convergence of geometry, bodies, tools, materials, and machines, in the language of computational abstraction. This convergence is fundamentally linked to the laborious definition—by a new social actor, the software engineer—of a new notation, a *code,* shaped by very concrete constraints: the size and materiality of the punched tape, the mechanics of milling, the geometric languages of part manufacturing, and the cognitive and bodily roles of machine-tool operators. Gradually, these commands and subroutines consolidated into increasingly abstract symbolic languages as the engineers sought greater efficiencies in machine-tool automation (see Figure 4).

5 Parsons had initially proposed a card-controlled Snyder milling machine for the project. However, as the Servo Lab researchers gained influence on the project, they chose to use a reconditioned government-surplus three-axis Cincinnatti Hydro-Tel milling machine provided (at no cost) by the Air Force.

As these languages became more general and versatile, turning general-purpose computers into special-purpose machines, the engineers' decisions concerning the representation of geometric, material, and machine constraints prompted the first theoretical formulations of the computer's role in design. The proto-software of paper tape for numerical control was indeed an embryonic theory of design, framing material, machine, and human operations—and, implicitly, a space of design possibilities. Thus, in contrast to the popular perception that computer-aided manufacturing is an offspring of computer-aided design, the opposite is true. The technologies, ethos, and vocabulary of manufacturing underlies the development of the first CAD systems.[6] However, with the subsequent focus on design automation as a research goal for the MIT engineers, and the pervasiveness of the metaphor of the digital as weightless and immaterial, most traces of the material and embodied origins of software automation would soon be forgotten. This needs addressing.[7]

Academy, Industry and Military Vectors Converging into Design

The project to automate a milling machine illustrates the entanglement of military, industrial, and academic interests in the U.S. Fueled by the awe-inspiring wartime advances in electronics, sensors, actuators, and cathode-ray tube monitors, a narrative of technological progress became prominent in the fabric of the U.S. identity during the Cold War era. As observed by historian Paul Edwards, during this period, a hegemonic view of technology optimistically portrayed computers as key to the U.S. national project of global supremacy and competitiveness, casting technology itself as a patriotic endeavor. Although computers were still unique, expensive artifacts—the privilege of an academic and government elite—the notion of widespread personal computing was already part of the popular imagination. The largest recipient of federal research funds during the post-war period—and the epicenter of a vibrant culture of technological research and development—MIT came to epitomize a spirit of engineering prowess of patriotic dimensions.

With this landscape of technological optimism as a background, the concept and technologies of numerical control—and later those of computer-aided design—emerged from army-sponsored research at MIT research laboratories. During and after the war, the U.S. Air Force was willing to fund projects to improve the production of components for military applications, such as airplane wings and helicopter rotaries. The Air Force motto, "MORE AIRFORCE PER DOLLAR"[8]—which was present in many of the research reports of this time—is powerful evidence of the militaristic roots of these efforts.

6 The precedence of numerical control over Computer-Aided Design is indisputable in light of the chronologies of the APT and CAD Projects at MIT, but links between the CAD Project at MIT and industry partners—such as Boeing and General Motors—through industry partnership programs, and the dynamics of adoption, may explain alternative views. Different views regarding the CAD-CAM lineage have been reported. For instance, a CAD/CAM instructor interviewed by Gary Downey contends that CAD did not originate from numerical control research, but from the need to manipulate very large amounts of drawings at Boeing. "In order to make the Boeing 747, they needed ten football fields of E-size [36 x 48 in] drawings. So CAD developed as a way of simplifying the drawing process." See Gary Lee Downey, *The Machine in Me: An Anthropologist Sits Among Computer Engineers* (New York and London: Routledge, 1998), 161.

7 For an extended discussion about the intellectual origins and institutional history of Numerical Control and Computer-Aided Design, see Daniel Cardoso Lach, *Builders of the Vision: Software and the Imagination of Design* (New York: Routledge 2015).

8 The all-capitals design appears in the original. Douglas Taylor Ross, *Investigations in Computer-Aided Design for Numerically Controlled Production: Interim Engineering Progress Report, December 1, 1963–May 30, 1964,* M.I.T. Report ESL-IR 221 (Cambridge, MA: Electronic Systems Laboratory, MIT, 1964), i.

The vision for numerical control did not come from MIT, but from John T. Parsons, then the vice president of the Parsons Corporation, Aircraft Division—an aircraft manufacturing company based in Michigan. Parsons sought the Servomechanisms Laboratory at MIT as a subcontractor to an Air Force contract he had obtained to produce a working prototype of the technology.[9] The Servomechanisms Laboratory occupied MIT's Building 32, a now demolished vast single-story warehouse building on Vassar Street. It was established by Gordon Brown in 1940, and housed in the Department of Electrical Engineering. In 1959 the laboratory changed its name to Electronic Systems Laboratory (ESL).[10] During the Second World War, the Servo Lab had focused on the application of servomechanisms for guided missile control and gunfire applications, and its personnel were key in the development of Project Whirlwind—a U.S. Navy effort resulting in the first interactive computer in 1946. Before contacting Brown, Parsons had secured Air Force funding to develop feasibility studies for an automated milling machine controlled by punched cards that was capable of producing aircraft parts; he had the idea of using servomechanisms to control the machine's movements along its three axes. An agreement was signed, and the collaboration between Parsons and MIT began formally in 1949, but Parsons's influence over the project diminished as the MIT researchers gradually took control of it. While Parsons, wary of costs, sought to fulfill his Air Force contract and develop a problem-specific application—a proof-of-concept device capable of producing wing panels for supersonic aircraft, the researchers sought to re-cast the project into a universal technology, promising a revolutionary transformation of manufacturing "applicable to any process which may be described in terms of code numbers."[11] As observed by historian David Noble, MIT's prestige, technical skill, connections with government, and proximity to Air Force sponsors were crucial for Servo Lab members to gradually displace Parsons's authority, to propose a course of action that exceeded the specifications of the original contract, and to ultimately secure a new contract directly with the Air Force (without Parsons's participation) in 1951 for the development of the numerically controlled milling machine. Changes in the language of the project's reports reflect this shift. While the first documents refer to "The Parsons Milling Machine," the final report, issued in 1952, refers simply to "The M.I.T. numerically controlled milling machine."[12] According to Noble, upon the project's completion, and despite multiple requests, Parsons was denied the project's technical details.[13]

Beyond its militaristic applications, a key objective of the project to automate machine tools was to disseminate the new technology to the manufacturing industry. Thus, MIT operated not only as a research powerhouse, but also as a broadcasting agency

9 According to several accounts, Parsons's interest in the MIT Servo Lab was spurred by one of his engineers, Robert H. Marsh, who was an MIT graduate and played an important role as a mediator between Parsons and MIT. See Robert H. Marsh, *An Evaluation of the Progress and Future Planning of the Parsons Milling Machine Project* (Parsons Corporation, January 31, 1950), AC 151, Series II, Box 28, MIT Archive. See also Noble, *Forces of Production,* 119, and Reintjes, *Numerical Control,* 16.

10 In 1978, the Laboratory changed again its name to Laboratory for Information and Decision Systems (LIDS). For a detailed history of the Servomechanisms Laboratory, see David A. Mindell, *Between Human and Machine: Feedback, Control, and Computing before Cybernetics* (Baltimore: The Johns Hopkins University Press, 2002).

11 Servomechanisms Laboratory, *Final Report on Construction and Initial Operation of a Numerically Controlled Milling Machine, Part I (Draft Copy),* Research Report (Cambridge, MA: MIT, 1952), 7.

12 Servomechanisms Laboratory, "Numerically Controlled Milling Machine Demo Announcement – Sep. 15, 1952," September 15, 1952, AC 151, Series II, Box 37, Demonstration September 1952, MIT Archives. Parson's exclusion has also been documented via interviews and additional archival materials by historian David Noble in Noble, *Forces of Production,* 109. A contrasting view is offered by MIT Professor J. Francis Reintjes, who was himself a member of the project in its later stage. Prefacing a detailed history of the numerical control research project, Reintjes argues that MIT's more general approach simply aligned better with the army's intent. He also posits that his account of this history has no drama because "there was none." See Reintjes, *Numerical Control*, xii.

13 Noble, *Forces of Production,* 131.

disseminating and promoting publicly funded technology. From MIT's perspective, this role aligned entirely with the Institute's educational mission. This alignment is illustrated by the public demonstration of the technology at MIT in a series of presentations in early Fall 1952. The three-day event was far from merely academic. James O. McDonough, one of the project's leading engineers, extended invitations to multiple army, aircraft, machine tool, and general industry actors. The attendance list was a "who's who" of Cold War-era corporate America, including executives, technical personnel and administrators from General Electric, the Munitions Board of the Department of Defense, Lockheed Aircraft Corp., Harvard College, and many others.[14] In addition to the live demonstrations and talks, hundreds of information packages were delivered to members of industry at large, and dozens of other organizations submitted letters of interest to gain access to the project's final report, which described the new technology as "a milling machine capable of manufacturing machined parts automatically by obeying a series of numerical instructions introduced into the machine on punched paper tape."[15] The program lasted all day and included discussions about "modern information processing, and numerical control," as well as demonstrations of the different stages of the new workflow: machine operation, tape preparation, and numerically controlled milling.

The 1940s and 1950s project of numerical control illustrates the convergence of military ideology and design discourse that persists to this day. The military's encouragement of an image of creativity and innovation linked to technologies of numerical control is clear enough. The Defense Advanced Research Projects Agency (DARPA) slogans—"to innovate we must *make*, to protect we must produce" and "Democratize Design" (italics mine)—illustrate the collapse, in public military discourse, of design and manufacturing technologies with national security imperatives.[16] The support is materialized in the agency's sponsorship of numerous civil initiatives, such as its funding of hundreds of "Makerspaces" in schools across the country, "Hackathons," as well as in its support of different "DIY" initiatives at both high schools and universities. This support is also explicit in mainstream political slogans—including presidential remarks—in support of the "maker" trope, and of specific technologies, such as 3-D printing.[17] With the ongoing promotion of these technologies through extensive programs of industry collaboration, academic publications, political discourses, and popular media, the boundaries of this technological enterprise become harder and harder to trace.

14 Servomechanisms Laboratory, "Numerically Controlled Milling Machine Demo Attendance List," September 15, 1952, AC 151, Series II, Box 37, Demonstration September 1952, MIT Archives.

15 Servomechanisms Laboratory, "Numerically Controlled Milling Machine Demo Announcement – Sep. 15, 1952," 2.

16 Dale Dougherty, "Makerspaces in Education and DARPA," *Makezine.com,* April 4, 2012, http://makezine.com/2012/04/04/makerspaces-in-education-and-darpa/ (accessed August 6, 2014); Maker Faire Bay Area 2012, "How DARPA Democratizes Design - FORA.tv," Fora.tv, 2012, http://fora.tv/2012/05/19/The_Next_Generation_How_DARPA_Democratizes_Design (accessed August 6, 2014); Anya Kamenetz, "Lasers, 3-D Printers, and Robots: The New Shop Class," Fastcoexist, September 18, 2012, http://www.fastcoexist.com/1680549/lasers-3-d-printers-and-robots-the-new-shop-class (accessed August 6, 2014). For insight into aspects of militarism in contemporary civil U.S. society, see Bryan Finki, Nick Sowers, and Javier Arbona, "DEMILIT," *Nick,* http://demilit.tumblr.com/ (accessed August 6, 2014).

17 Barack Obama, "Remarks by the President in the State of the Union Address," The White House, February 12, 2013, http://www.whitehouse.gov/node/197846 (accessed January 31, 2015).

The Rise of the Gentleman Technologist

The builders of this vision of a manufacturing revolution via numerical control—the MIT engineers Douglas Ross, Gordon S. Brown, Jay Forrester, James O. McDonough, and others—are representative of the twentieth century emergence of a new social figure in the United States. I call this figure the "gentleman technologist." His command over technological systems grants him a special place of authority in society—away from the toil of the machine shop and closer to the spheres of power. The gradual displacement of authority from Parsons to the MIT engineers signals the arrival of this new, typically male, figure on the stage. While John T. Parsons had the idea of controlling machine tools with punched media, control of the project—and many of its benefits—went to the MIT engineers who developed, implemented, and later patented the system based on his idea.[18]

This aura of technology did not shelter those who, merely a century or so earlier, were invested in the mechanical arts—technology's cultural and historical predecessor. To understand this shift we can compare the Parsons-MIT conflict with another, earlier conflict in the history of technology: that of nineteenth century computing pioneer Charles Babbage with his engineer, Joseph Clement, over the intellectual ownership of the Difference Engine—a mechanical predecessor to modern computers that normative histories of computing attribute solely to Babbage. Clement, a talented engineer and draftsman who built the machine, bitterly and unsuccessfully disputed Babbage's authorship of the device. As observed by historian of science Simon Schaffer, in the Babbage-Clement dispute the place of intelligence itself was at stake: Is it located in technology's "conception," or is it in its "making"?[19] The design of modern computing technologies indexes a shift in this long-standing struggle for authority and control. Notably, in contrast to Clement's unsuccessful claim to credit for building the machine, the engineers at the Servomechanisms Laboratory succeeded in obtaining most of the reputational and commercial benefits derived from making numerical control technologies.

Leaving the politics of technological authorship aside, we can usefully focus on the specific nature of these gentlemen technologists' achievements. As we shall see, the invention of the technology of numerical control, based on the codification of geometric, material, and machine constraints, was fundamentally shaped by the materiality of available substrates and by a new set of skills required for their manipulation. Computational abstractions are in essence material and their development requires a special kind of craftsmanship.

18 The MIT engineers, headed by Jay Forrester, filed a patent in 1952 for their punch card system. It was awarded in 1962.

19 Simon Schaffer, "Babbage's Intelligence: Calculating Engines and the Factory System," *Critical Inquiry* 21, no. 1 (October 1, 1994): 203–27.

From Shop to Code

In automating machine tools, numerical control researchers sought to replace an individual's embodied engagement with a machine—the trace operator's—with a repeatable and controllable digital process akin to a symbolic calculation. Reinventing material manipulation itself as computation was aligned with an ideology of automation and total control seeking to precisely manage—and reduce—the involvement of humans in processes of production. These researchers, and their military sponsors, imagined that reducing the steps between design and manufacturing would result in a cleaner, more efficient process. Again, the U.S. Air Force motto, "More Airforce Per Dollar," comes to mind. However, the technologists' attempt to codify the manual craft of operating a milling machine through traces resulted in a complex socio-technical system that demanded different skills, and in a different kind of craft that, instead of reducing work, transformed it and relocated it from the shop to the programmer's desk.

When the engineers at the Servomechanisms Laboratory completed the construction of the first numerically controlled milling machine in Spring 1952, they hailed the new development with claims of efficiency, freedom of human error, and announcements of a manufacturing revolution. However, producing the information and punching it into tape was still an arduous manual process involving long hours of complex calculations to encode a part's geometry in mathematical form, to calculate the movements of the machine's three axes, and to account for variables such as cutter type, size, speeds, and the sequence of cut operations. When done manually, or with the help of electro-mechanical desk calculators, this planning stage could take several hours for a very simple shape. Once the machine instructions were ready, the "part programmer" would give them to the "keypuncher," who used an eponymous device to produce the machine-ready paper tape. A Servo Lab insider, Professor J. Francis Reintjes, recalls that "machining efficiency came at the expense of time consumed in programming for that efficiency."[20]

Confronting this problem, the engineers sought to automate the production of machine instructions, first by writing subroutines encoding commands for particular profiles, thus saving time, and later by creating higher-level problem-oriented languages allowing for more flexibility in the "job planning" process. The earliest attempts to automate the production of machine instructions can be traced to the work of John H. Runyon and Arnold Siegel, who worked on the Whirlwind computer.[21] But the engineers quickly realized that job planning demanded higher-level abstractions. To address this need, the engineers sought to build a

20 As the director of the then-Servomechanisms Laboratory beginning in 1953, Professor Reintjes participated in the development, led by Douglas T. Ross, of the APT. Although Reintjes left the ESL in 1960, an important part of the project evolved within the context he helped to create, and thus his accounts are of great value. Reintjes, *Numerical Control,* 54. Under his direction, the laboratory became a melting pot for graduate students, faculty, and researchers in a diverse array of fields, including "hard" sciences like mathematics, physics, and electrical and mechanical engineering, as well as seemingly distant fields, such as chemical engineering and food science. MIT News Office, "Professor Emeritus J. Francis Reintjes Dies at 96," *MIT's News Office,* March 5, 2008, http://web.mit.edu/newsoffice/2008/obit-reintjes-tt0305.html (accessed March 12, 2014).

21 See John H. Runyon, *Whirlwind I Routines for Computations for the M.I.T. Numerically Controlled Milling Machine* (Cambridge, MA: MIT Servomechanisms Laboratory, 1953).

scalable language for machine tool path specification that would allow users with no programming skills to use numerically controlled machine tools. The development of this language began in 1956 under the leadership of Douglas T. Ross, a young mathematician who had worked on a flight simulator in the Servomechanisms Laboratory—but who had no experience in either design or manufacturing. "From the computer application's point of view," he wrote years later, "the primary problem is not how to solve problems but how to state them."[22] Through an aggressive program of dissemination and industry collaboration, this language, named Automated Programming Tool (APT), would become a worldwide standard for the aircraft industry in 1978. The following list is an excerpt from the APT dictionary from 1958, illustrating some of the language's key commands:

APT WORD	MEANING
ALL	Plot all cutter coordinates.
AT ANGL	At a specific angle from the positive X-axis.
AUTO	Automatic.
CENTER	Center of a conic section or sphere.
CCLW	Counter clock-wise.
CLW	Clockwise.
CROSS	Cross product of two vectors.
ENDARC	Defines the end angle in degrees.
FULL	Full Plot.
FUNOFY	Function of Y.
INTOF	Intersection of.
LEFT	Designates the left hand side, looking in the direction specified by method of geometric definition.

While his predecessors at the Servomechanisms Laboratory had been concerned with spindle speeds, data commands, and part geometry, Ross pondered how to represent points, lines, problems, and even complex artifacts, such as houses and circuits, and even language itself. A trained mathematician, Ross's fundamental concern was representation. His ambition was to seek a general codification system—a universal language. This ambition reaches its apex in the *plex*, a theoretical construct he defined, esoterically, as "an interweaved combination of parts in a structure… [with the purpose of representing a] thing, be it concrete or abstract, physical or conceptual."[23] Ross imagined the *plex* as an all-purpose representational unit—in fact, with philosophical implications. To represent a line, for example, a *plex* had to be defined as that which contained sub-entities for its starting and ending points; each

22 Douglas T. Ross and John Erwin Ward, "Investigations in Computer-Aided Design for Numerically Controlled Production: Final Technical Report" (Electronic Systems Laboratory, Electrical Engineering Dept., Massachusetts Institute of Technology, May 3, 1967), 175, MIT Archives.

23 Ross defined the *plex* as the combination of three key components: *data, structure,* and *algorithm.* The *data* are "units or indivisible entities in terms of which the 'thing's' properties are described or measured." The *structure* refers to the relationships between the data, and the *algorithm* is "the capstone that allows the data in the structure to be interpreted, manipulated and filled with *meaning."* The algorithm relates to the behavior and the interpretation of the whole: a sort of logical rule set for operation and assembly. With the *plex*, Ross sought to create a general theory of representation for describing (and computing solutions to) *any* problem. Whether the artifact to be designed was a servomechanism or a house was irrelevant. See Douglas Taylor Ross, *Investigations in Computer-Aided Design for Numerically Controlled Production,* Report ESL-FR 351 (Cambridge, MA: Electronic Systems Laboratory, Electrical Engineering Dept., Massachusetts Institute of Technology, 1968), 13–15. http://dspace.mit.edu/bitstream/handle/1721.1/755/FR-0351-19563962.pdf.txt;jsessionid=3469E7BE3780EDAF65F833757A012AF4?sequence=2 (accessed July 17, 2014).

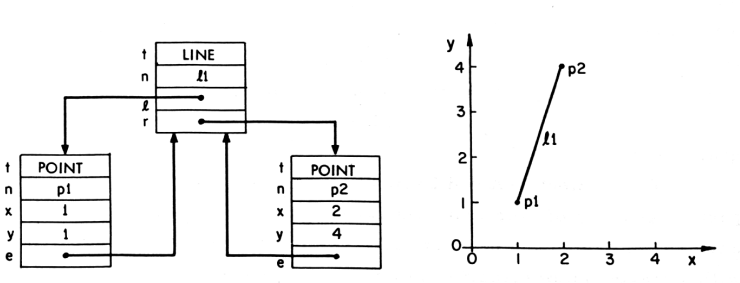

Figure 5
Douglas Ross' Plex. Rights: *Permission granted by Massachusetts Institute of Technology.*

point sub-entity would in turn contain values for its *x* and *y* coordinates (see Figure 5). Another sub-entity would describe the line itself as an independent element with pointers to the other two sub-entities.[24] Aligned with contemporary interest in artificial intelligence, Ross imagined *plexes* as user-definable, interpretable, and computable. An interpretive system would transform the user's verbal or graphic representations into "internal models" with which the system could compute—a self-consistent universe of interacting "meanings," opaque to an external observer—a *black box*.[25] Crucially, the purpose of this representational and interpretive apparatus was to enable the automation of aspects of design. In a 1959 letter to a Ford Motors executive, Ross makes this desire explicit: "One of our main interests will be to attempt to increase the language capability for communicating with automatic programming systems of this type, and also to attempt to automate some of the design process itself."[26]

The engineers at Servo Lab not only thought that design could be represented symbolically through *code*, but also automated. Ross sought to implement aspects of this theory in what arguably is the first attempt to automate aspects of design—the Automated Engineering Design (AED) programming language. The AED effort, as well as its APT and *plex* predecessors, shows how these technologists viewed design as a problem of representation and codification—a question for which language building was an appropriate answer. AED was in fact the language of choice for a pioneering work of CAD. MIT Professor Emeritus of Architecture William L. Porter used it in his 1968 dissertation to create a system for generating urban design alternatives in a variety of scenarios. Porter recalls that, unlike other available languages, AED seemed to offer the possibility of "declaring meaningful statements about design."[27]

More generally, despite its esoteric formulation, Ross's *plex* exposed a theoretical commitment to the idea that computational descriptions, because of their capacity to index data, can be computed with (and bear structural resemblances with) the artifacts they are meant to depict. If we can compute using abstractions of real-world design situations, the *plex* logic goes, then design practice itself could be routinized. Engineers, then, did not merely

24 The dissociation between data, structure, and algorithm, explicit in the *plex,* is in a way essential for the programming of graphic representation systems. From this perspective, the *plex* theory suggests the imminent appearance of object-oriented programming (OOP). MIT doctoral student Ivan Sutherland's Sketchpad, widely recognized as the first interactive computer graphics system, is also widely considered the first example of OOP—an achievement for which Ross would claim credit afterward. See Sutherland, Ivan Edward, *Sketchpad, a Man-Machine Graphical Communication System*, Massachusetts Institute of Technology, (Cambridge, MA: Massachusetts Institute of Technology 1963).

25 In computing, the expression "black box" refers to a system whose workings are opaque to an observer. In computation, it's a common expression used to refer to aspects of a system that are beyond the reach of a user. The following quote, from a paper titled "Theoretical Foundations for the Computer-Aided Design System" illustrates these notions of self-containment and opacity: "Since the entire process is based ultimately upon the interactions between the meanings of the many elements involved, and since the sorting out of what things go together is handled automatically by the 'natural laws' of behavior which are built in, the designer on the outside has no conception of the chaotic activity inside the system, but sees only external effects appropriate to his mode of understanding." See Douglas T. Ross and Jorge E. Rodriguez, "Theoretical Foundations for the Computer-Aided Design System," in *Proceedings of the May 21–23, 1963, Spring Joint Computer Conference*, AFIPS '63 (Spring) (New York: ACM, 1963), 318.

26 Douglas T. Ross, Letter from D.T. Ross to G. Pascoe, January 22, 1959, AC 151, Series II, Box 36, Project 683, Correspondence AIA, 1959, MIT Archives.

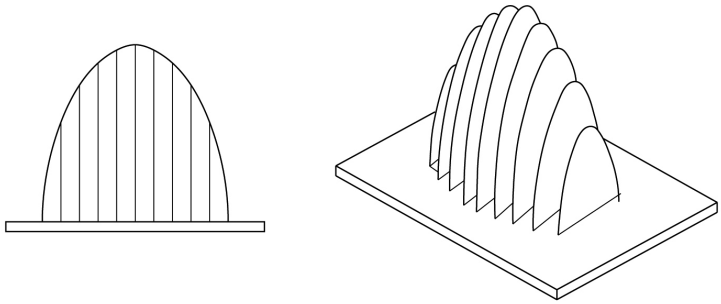

Figure 5
Reconstruction of an APT drawing.
Author owns rights to this image.

automate the manual work of trace-controlled machine operators; they transformed it, uncovering new problems and opportunities. The incremental-coordinate continuous-path notation system developed by McDonough and Pease embodies not only an appropriation but also an algorithmic re-interpretation of manual work (see Figure 6). As these examples show, software does not merely re-create sites of practice, but actually transforms them—and in the process generates new difficulties, as well as necessities for new forms of labor and skill: Notations had to be devised so that machine tool commands fit efficiently in the paper tape; the process of planning a part for automated machining had to be made less laborious than the trace-controlled operations of the past. Beyond their partial implementation into operable software systems, the *plex* and the APT and AED efforts synthesize a fledgling philosophy of design and manufacturing linked to software construction; it sought to take problems from the messy worlds of materials into the "clean" worlds of symbolic abstraction—from shop to code.

The Place of Design

Technological systems index their makers' theories of action, thus modeling users, machines, and their interactions. As anthropologist of science and technology Lucy Suchman notes, technologies can be seen as "propositions for a geography where relevant subjects may claim their place."[28] If we are to use this lens to interrogate the MIT project or machine tool and design automation, what are the notions of making, of designing, and of the human that it indexes? Where does it (re)locate design's people and practices? Servo Lab engineers clearly pondered the role of human intervention in the new automated environments for design and material production. Their new languages and devices demanded new skills and literacies, shifting the social and technical place of design. We can see this shift in the new design and manufacturing workflows envisioned by numerical control researchers.

Drawing inspiration from then-contemporary cybernetic discourses, which construed human–machine interaction as a symbiosis between two organisms, numerical control engineers reimagined the designer's role in relation to what they perceived to

27 From an e-mail conversation with the author. William Lyman Porter, "Re: Question about DISCOURSE and the AED Language," June 29, 2014. For the complete thesis, see William Lyman Porter, "The Development of DISCOURSE: A Language for Computer Assisted City Design," Thesis, (1969), http://dspace.mit.edu/handle/1721.1/39037 (accessed August 18, 2014).

28 Lucy Suchman, *Human-Machine Reconfigurations: Plans and Situated Actions,* 2nd ed. (New York: Cambridge University Press, 2006).

be the machine's capacities.[29] In their cybernetic theories of designing and making, they assigned complementary roles to each part of the human–machine assemblage, defining the boundaries of what constitutes *creative* work.[30] Key to these workflows was the idea that the distance between design and production could be reduced—collapsed, even—through automation. The laboratory's reports and memoranda are rich with diagrams and rhetoric depicting design-to-manufacturing scenarios where humans are progressively replaced by technologies. Here, the replacement of the human is presented not only as the optimization of an industrial process, but also as a form of emancipation: a way to "free" humans from the toil of dealing with materials and to "liberate" them as creative agents. Computers became, in the imagination of these gentlemen technologists, humans' *perfect slaves.*

Taking this image of gradual automation to its natural conclusion, design and manufacturing are to be performed by an individual human mind that directly and disembodiedly commands machines to materialize designs—a desire that resonates in current discourses of digital fabrication centered on the presumed immediacy and seamlessness of rapid prototyping and 3-D printing. Freed from the toil of manufacturing and calculations, this image suggests, humans could devote their time to "creative" endeavors. And yet, as we have seen, the laborious design, development and maintenance of the symbolic languages and sociotechnical infrastructures that support these digital transactions, the demand for new skills and social roles, and the often-messy operation of machines, complicates this dominant image of technology in design. Contrary to pervasive narratives of the digital, computational transactions are not invisible, nor weightless. They are socially and materially constituted. As theory and contract of such transactions, software—its history, its design, its irreducibly material dimension—must be approached as a crucial subject of discussion and debate in design.

29 The science of cybernetics, first formulated by the mathematician Norbert Wiener, conceptualized biological, mathematical, social, and mechanical systems as flows of messages and feedback loops, susceptible to control. Stemming from the wartime advances in servomechanisms and control, this new scientific paradigm framed the efforts of numerical control and computer-aided design researchers. Licklider's article, "Man–Computer Symbiosis," and Shannon and Weaver's information theory were particularly influential. See J.C.R. Licklider, "Man–Computer Symbiosis," 1960, http://groups.csail.mit.edu/medg/people/psz/Licklider.html (June 3, 2014) and Claude E Shannon and Warren Weaver, *The Mathematical Theory of Communication* (Urbana and Chicago: University of Illinois Press, 1998). See also Norbert Wiener, *Cybernetics, or Control and Communication in the Animal and the Machine,* 2nd ed. (Cambridge, MA: The MIT Press, 1965).

30 Ross explains: "The synergetic integration of the creative abilities of the human with the immense capacity of hardware and software in the computer, in a man–machine problem-solving team." Ross, Investigations in Computer-Aided Design for Numerically Controlled Production.