What Do We Mean by “Program”?
The Convergence of Architecture and Interface Design

Benjamin H. Bratton
Yahoo!, SCI_Arc, UCLA | benjamin@bratton.info

Very often in my discussions with interaction designers, there is strong agreement that their work should include in its assignment a broader range of interactions, including things like supply-chain systems, recycling techniques, intangible workflow processes, traffic interfaces, and even furniture. There is a hope that their discipline would have something important to say (and do) for the design of interaction wherever it may meaningfully occur. There is, however, less insight than enthusiasm into how other design disciplines have thought about these issues, and therefore where closer bonds are most needed. If interaction design is to make good on its aspiration to move beyond the relatively closed, localized experience of clickable software buttons and expand into the more open, expansive agenda of how human-scale and city-scale interactions can be designed, then it will do well to consider how architecture relates to human-machine interaction (as body and building) and what might be learned from it.

Design is not only the design of things in themselves but perhaps more important the design of how things work together (something more complex and elusive). Architecture refers to this working together as “program”—a set of designed or designable scripts that organize organization itself, that imagine in advance how things will play out, and stage their interrelations accordingly. It seems that the word and the concept have a vital and specific role in enabling the necessary scalar shifts to which interaction design aspires, one which is not only about the convergences of interfaces (and disciplines) but also their replications and divergences.

Embedded Interfaces: Software and Space
To investigate what organization-of-organization really means, we have to first specify what it is that programs program, namely “interfaces.” If we think of interface design as an expertise in the points of contact between complex and perhaps incompatible systems, the design of interfaces has everything and nothing to do with software or architecture per se, as both systems are themselves organizations of interfaces programmed in particular ways. We study and produce interfaces between humans and computers but understand that the chain of interactive succession extends far beyond the moment when buttons on screens are clicked. It extends all the way down into global and local networks of systemic interconnectivity, including and dependent upon concrete, tangibly embedded interfaces like buildings, cables, and cities. Conversely, once we think about how and why physical things are moving as they are through urban space, we discover the software under the surfaces of everyday hardwares. Each couches the other.

I hope to clarify—particularly for interaction designers who specialize in the creation of software graphical user interfaces—a necessary convergence between these two nominally distinct design practices, architecture and interaction design, and to sketch an image for programmatic thinking that is more encompassing than has been previously considered. My hope is that as points of contact between complex systems are necessarily both physical and virtual, and as each creeps further into the domain of the other, a combined agenda of architecture and interaction design will emerge called perhaps simply “interface design.” I would hope that this emergent discipline will be widely understood not only as a design craft but also as having significant
ramifications for organizational management and governmental theory and practice.

In the concentration of architecture and interaction design, this interface design will have less to do with the formal correspondence of software that uses architectural metaphors (such as Second Life) or architecture that uses software metaphors (such as airport signage), or even “interactive architecture” (think Jacques Tati at Ars Electronica) than pragmatic convergences and divergences of the social and cultural operations and expressions they all share: the primacy of the idea and ideal of “program.”

**Program Per Se**
**Propositional Infrastructures**
Where do programs come from and why? Program is the image (prescriptive, diagnostic, analytic) of how a field of interfaces comes together into a specific active system. When architects are engaged in conceiving the integration of structure, space, layout, partition, void, pathway, of how people course through them and push against them, and of how those structures and pathways will push against the people, they call this thinking and planning “programming.” The design of architectural programs, like interaction design for software, is based in the creative analysis of organizational needs and possibilities and crafting an infrastructure to best stage desired conditions of connection and interaction. In the most specific sense, the architectural discourses around “program” (and there are many, both enthusiastic and dismissive) are where the question of spatial interaction design comes into play, even if it is seldom referred to as such.

In architecture, program is the framing script for how inhabitants will engage with a spatial system over time, or over a day, or simply from one place to the next. It is the prescriptive imaginary for how the architectural space will coperform its functions with those people and things that it will house. Hospitals are planned in a certain way because medical programs demand it. Prisons have regimented partitions and interfaces because the penal discipline uses these as its physical methods of punishment, control, and surveillance. Warehouses are planned and coded as they are to support the efficient movement and display of transient inventory, a program of logistics. A border between two countries is put together the way it is in response to the discourses of the nation and its designed contiguity, as well as security interfaces. Even more dynamic, culturally specific environments utilize program: Successful carnivals (and protests) opportunistically take advantage of the urban environment to upset everyday work and function. Program is not just about function. It is more about setting up the terms (designing) by which the built environment (including hardware and software) can actively participate well in social and economic intercourse and then letting life assume the territory.

In the conventional sense of architecture, interfaces converge in a fixed site bound by a building envelope, creating in their arrangement a bound interior. Today our own hypermodernity can make some stable systems (such as libraries and nations) more fluid and liquid, and the volatility and divergences of interfaces make the joint agenda of architects and interaction designers that much more critical.

**Hard and Soft Interaction in Historical Context**
For this expanded landscape, in which soft and hard systems intermingle on the go, the concern of architectural program is still to stage desired conditions of connection and interaction,
separation, and discreteness, but it is as much about network effects and affects, software infrastructures, and experiences, as it is about glass and concrete. The blending of architectural thinking and media-information technology design is by no means new. Quite the opposite. But its story tends to be told from the perspective of architecture, the more discursively mature discipline. Interaction design as a discipline does not have a coherent, historical discourse. It does not, as of yet, have all that much to say about how “interaction” was designed in ancient Rome, in the midst of the Paris Commune, or among the Iroquois confederacy. While today interaction design is more practical than epistemological, this will likely not be the case by the middle of this century. As computer science and philosophy annex each other, interaction design may do something similar with sociology and architecture. If so, this may prove less the end, vanishing, or eclipse of architecture (as some have suggested) than the disclosure (finally!) of its first truly post-Industrial assignment.

As we look forward to a design practice with holistic concern for all kinds of human-related interfaces, the lessons learned about the problem of “program” by architecture are of direct value to interaction designers who are now asked to solve similar organizational design briefs with different tools.

**Architectural Program in Transition**

How has the notion of program changed as architecture’s own image of itself has changed? How has it driven and been driven by cycles of modernity and media technologies?

First, program needs to be understood in relation to a reflexively generative relationship between bodies and the physical spaces they inhabit. French sociologists Pierre Bourdieu and Henri Lefebvre employed the notion of “habitus” to analyze this very reflexivity. Habitus is the root from which both habit (as in bodily habit) and habitat are derived. It can be used to discuss how one informs the other, how bodily habits, especially of large populations, wear grooves into space, producing habitats in their image. For one, this is how archaeologists are able to divine social practices from reconstructed architectural debris: The forms imply what made them. Conversely, space frames and constrains the action that it houses, training bodies and thereby program in its image. To be sure, habit and habitat emerge at once, a conjoined machine, not in some never-ending representational reciprocity. In essence, the patterns that program seeks to specify can be understood as the prescriptive or analytical images of this emergence.

In the early years of modernity, architectural programs may be seen as explicitly diagrammatic of a larger ethos of moral order. Usually drawn as plans (two-dimensional line-based images of a site, as if seen from above with the roof removed), schemes relied heavily on graphical symmetry to imply and ensure the clear-mindedness of their implications. (Examples of this include Charles Fourier’s plans for a utopian community, or Michel Foucault’s analysis of Jeremy Bentham’s ideal prison.) In these a recommended moral order is drawn into an unfolding embodiment of adjacency, hierarchy, discipline, and procession. The architecture that would be constructed is almost directly translated and extruded from these idealized, floating arrangements. This configuration was well-suited to a society in which large, immobile institutions were constructed to transfer scientific and moral authority onto a relatively local, immobile population. Though to this day, the plan remains the preferred device with which to draw program; for it the “Floor” is the tablet to be partitioned by lines, and program is modeled as a compartmentalization of habitat type into clear zones of activity.

But modernity’s accelerations brought radical forms of urbanization, technological speed, and rationality, as well as new mobilities of people, labor, and capital. Traditional social forms were uprooted, populations were moved to city centers, and sidewalks filled with strangers, unskilled and hyperskilled segmented into increasingly narrow specialties. Places of work, factories or high-rise offices, where those modern interfaces converged and synthesized, became machines of functionalism. Reciprocity developed: Architecture and industrial design sought to train an organization into the rational image of its new equipment and simultaneously to model that equipment in the image of an ideal organizational form. The social agenda was not only to accommodate, but continu-
ing the modernist project, also to improve it according to its own abstract ideals. Form itself was hyperrationalized, reduced to its most essential purpose. Calculated, or so it was claimed, for maximum performance and throughput. "Interaction design" proper was born, both sinister and progressive (in the image of both Taylor’s studies to minimize assembly-line movements and Louis Kahn’s studies of traffic patterns in Philadelphia). Formal aesthetics also emerged that mirrored this machine logic; functionalism became codified into design ethics of usability and later with it the formal application of persona typologies as design-research techniques.

But the machinic mechanization of programmatic efficiency was also in conflict with the cultural ethos of postwar society and its celebration of the individual. The question of who was using whom became increasingly more volatile, exploding even in the world’s cities in the late 1960s in a broad social and political revolt against the very logic of efficient function that opaque, rectilinear modernist corporate architecture embodied. Function became the enemy of possibility, grided structure the opposite of open innovation. And against this backdrop, "program" became a dirty word. Its projective logic not only remained, however, but it also blossomed, less a script for organizational efficiency than a platform for speculative conceptualization of impossible alternative cities.

**Challenges for Architectural Program in a Software Society**

And so today, in facing the "Google economy" and the challenges of a hyperglobalized, increasingly mobile, and intensely capitalized economy, architecture finds itself confronting several paths and roadblocks at once, some formal, some genetic, some informational, some phenomenological, some performative—some having little to do with buildings as we might normally think of them.

Architecture critic Hans Ibelings describes architecture’s programmatic responses to millennial globalization as “supermodernism.” He describes how the rate of change of occupancy, the requirements of flux and flow, the legal and financial constraints of the buildout, not to mention the sheer velocity with which information pushes people and goods through volumes and voids, has focused architecture’s expertise (1) to the blunt envelope’s requirements of a hyperperformative of high-performance spaces and superbrutalism (as volume and wedge) and (2) to the maximal intelligent, intricate, and interactive surfaces (as skin and screen).

In this, formally specific (and therefore static) programmatic solutions underestimate the contemporary speed of space and are replaced by cheaper, more flexible dumb sheds. Mutability of occupation through quarterly financial cycles wins out over specific frameworks, which become so much noise in the system and end up contributing friction and inefficiency. Anything that reduces flexibility that is anything but smooth (literally and metaphorically) is only decorative and unassimilated into the larger network of space, capital, and experience. Think of the difference between

Today architecture finds itself confronting several paths and roadblocks at once, some formal, some genetic, some informational, some performative—some having little to do with buildings as we might normally think of them.
the singular Chrysler Building in New York City, a permanent public exoskeleton of the corporate body casting up into the sky, and on the flip side a network of giant metal rectangles in the middle of a field in Arkansas where cars are today assembled within jumbo shipping containers in economic and political obscurity.

But this is only one response to the requirements of instrumental interfaces. And so the architecture story is actually a software story as well. Below I consider convergences, replications, and divergences in more detail.

**New Context: Architectural Interfaces Becoming Software Interfaces**

Architecture is being transposed, translated into software systems. Techniques of organizational management (where it is that bodies-in-space go, and what they do once they are put there, how they come in contact with each other) that were once the expert province of architecture have since the mid-1960s been transferred to increasingly powerful and nimble software-hardware complexes. This has had a considerable effect on the push and pull of corporate centralization and decentralization, social nearness and farness, cultural privacy and publicity, and individual and collective participation in the modern organization. Much interaction design done during this period sought to provide clearer interfaces between individuals in this organization to the software, between individuals and individuals through the software, and between some software and other software.

**New Context: Software Interfaces Becoming Architectural Interfaces**

Does this transposition mean that in turn, software becomes the architecture? Yes and no. In the 1990s it was fashion-
ture allows for better connectivity (functional proximity) between Los Angeles and New York than, say, Los Angeles and San Bernardino. So as the proximity to rivers, oceans, and shipping ports sparked commerce in the past, today proximity to cheap, plentiful, open bandwidth becomes the geographically specific interface upon which organizations rely to centralize their operations. This produces vastly complicated intersections of politics, economics, and social culture, both intensifying links between some territories and further disconnecting other regions. In this, the spatial career of software is entirely dependent upon large-scale architecture as its medium and force, and vice versa. It doesn’t displace the meta-interface of the city—it feeds it.

Information technologies and social systems of spatial formation, interaction, signification, emergence, and complexity always commingle and codetermine each other. Today software is an increasingly promiscuous substance. It intermingles with every object. It either animates that object (like a gasoline pump, a street light or an alarm clock) or it guides that object to us (like a gasoline pump, a street light or an alarm clock). This displace the meta-interface of the city. Today software is entirely dependent upon large-scale architecture as its medium and force, and vice versa. It doesn’t displace the meta-interface of the city—it feeds it.

Computation in the Wild
Further, for most of the world’s citizens, their first owned, personal computer will not be a (hundred-dollar) laptop but a (ten-dollar) handheld device/phone/PDA of some kind. This is a critical difference in the experience of world-software interactions. For the West, the scalar phenomenology of computing runs from large to small, from architecture to accessory, from building-scale mainframes to desk-scale PCs to lap-scale laptops to hand-scale PDAs. Similarly, our understood history of telephony is one that moves from fixed-line phones (where telephony was a fixed function of the architecture) to one untethered within the domain of domestic space, to finally one that we put in our pockets. Phones were “de-architecturalized”; they become an extension of our bodies, not our homes and offices. As the modern city was built around providing real and virtual proximity, shared space and centralized bandwidth, a new city is emerging that two billion people already carry in their hands. For most humans (the “we” that live in “developing countries”), the personal experience of computational information media will start off as hypermobile, conversation-capable devices. For most, computation will begin as a sort of remote control for the world, albeit one centered around synchronous P2P voice protocols (aka phone calls). Because of this, the really important applications in pervasive computing (those built on emergent social behavior around information, and those that enable the acceleration of informal economies to connect across urban, political divides) will more likely emerge from internet cafes and third bedrooms in Delhi, Dakar, or Dar Es Salaam than in VC-funded cube farms in Sunnyvale, Cambridge, or Chelsea. At least one can hope so, and for them to separate interface design into hard and soft touchpoints will be less meaningful.

Program and Interface Design: A Heuristic
Program, regardless of medium, is always a translation of possibilities in relation to a set of contexts, themselves never directly given, rather filtered, interpreted, interpolated by a design scheme in particular ways. What appears obvious or even universal in a point in time to one design logic in its translation of context to solve a particular “program” (library, school, house, etc.) is completely unintelligible and bizarre to another logic at another time or place. This is no different for the design of software or any other interface.

What would integrated look like? What is common language for this, a common language for the specification of soft and hard programs? Both disciplines are tasked with designing how complex organizations come in contact with their own people and processes, with external
organizations and systems. Both are critically concerned with solutions that provide continuity and structure on the one hand and fluidity and iteration on the other. Both aspire to model these for spatially embedded interfaces, mobile interfaces on the go, for one-way information, two-way communication, for search and location. Both are projective in their purpose: Both imagine an organization performing differently, as a different organism, and both install hard and soft structures to bring that about.

**Universal Interface Design**

This is rudimentary information architectural theory, or should be. Again, “program” is the image of how interfaces, of whatever kind, come together into a real or potential active system of transference and communication. As a design venture, program is projective, even speculative. What will happen with the introduction or removal of some thing or process? What will happen to organizational translation? What is replaced or displaced in this? What intervention is at work in habit and habitat? This may begin as an analytical process, full of persona and testing, but no amount of research will add up to an irreducibly risky act of projection, to a simple act of design.

This is where software “becoming” architecture means not only software becoming a kind of environmental intelligence, but also that the conceptual design of software becomes, like the architectural imaginary, a discourse for the projective conception of what our shared environment could be and should be. This is clearly already at work and is to be encouraged as such. But for its part, software design is still underestimated as tactical, as a technique of management rather than management itself. It is not merely solving a particular problem, any more than a building is merely solving a problem of space. Both are strategic, projective, and innovative.

But software, strange as it may sound, perhaps has still to clarify the horizon programs available to it. The politics of FL/OSS is critically important as a model for democratic infrastructure, but it is limited as a politics of code and code developers. Privacy is important, encryption tools are important, open bandwidth is important, access for the disadvantaged is important, but these are more definitions of a level playing field than designs on what beautiful things might happen on that field. As software comes to animate space directly, a more universal interface design is necessary (and to great extent this is how I characterize my work to people confused as to why a sociologist is so concerned with architecture and software in the same breath).

Part of the development of more ambitious and omnivorous programs for interface design is to extend the specification of what hard and soft programs actually do, to draw the design profile of software not by its size or its application purpose, or its place within a stack, but according to its situation within a larger organizational habitus. What does it—as a complex of interfaces—converge, replicate, diverge, and when, and how does it do this? Are those results transferable to other contexts, or not? Why not?

I end with an assignment for the reader: Design a political border. We know that it is an interface between two complex systems. But how big or small, strategic or tactical is the interface-design assignment there? It is an architectural assignment (as a recent New York Times competition demonstrated), but it is also a software design problem. Maybe it’s just a matter of smoother logistics of the same basic processes, or maybe the possibility is much more open than that. Next week some young visionary interface designer, not knowing that hard and soft systems are supposed to be dealt with by different professionals, may posit that geographic borders—lines in a planometric vision of terra firma implying contiguity where there is none—are incompatible with her program for how politics, cartography, and security should be constructed. When she does I will be listening carefully.

**About the Author**

Benjamin H. Bratton is a sociologist based in Los Angeles who specializes in design economies, media theory, architectural theory, and software studies. He is director of the Yahoo! Advanced Strategies Group, where he develops next-generation search, brand, and interaction platforms. He is also the senior cultural studies faculty at SCI_Arc (The Southern California Institute of Architecture), lecturer in the Department of Design/Media Arts at UCLA, where he co-directs the Brand Lab, and a founding fellow at the CALIT2 Center for Software Studies at U.C. San Diego. Visit him at www.bratton.info.