Shift+Design: Scripts and Other Design Artifacts

Daniel Cardoso Llach
ABSTRACT

Beyond the debate about the possible advantages of the automation of time-consuming drafting tasks, or of the expressive qualities that emerge from the use of scripts in architectural design, this article posits the idea that scripts constitute a new kind of “design artifact”, reconfiguring a designer’s engagement with a design problem. By examining how scripts destabilize dominant conceptions of architectural representations as figural descriptions the article delineates the ongoing emergence of a performative and computational epistemology of architectural design.
1. Introduction

This article reports on the author’s work with architecture students during a series of scripting workshops across different countries and institutions. It is aimed at describing how both the scripts and the stories woven around them reveal conceptions of design constitutive of novel design identities, in which notions of design, architecture and creativity are rephrased in computational terms. The workshops were conducted by the author in collaboration with Kenfield Griffith, John Snavely and Skylar Tibbits.

Architecture schools are sites of epistemological conflict, of peculiar tension between different disclosures of the form vs. content conundrum. Most schools today place curricular emphasis on design studios while other areas, deemed more “technical” or “theoretical”, are rendered subsidiary. To
seek to situate scripts within the physical and conceptual space of the architectural studio the first section of the article discusses comparatively aspects of architectural and computational pedagogy. The seemingly conflicting ethos of these two is discussed under the light of recent advances in the social sciences, placing emphasis on the commonalities between architectural design and computer programming.

The second section discusses the experience of two teams of architecture students engaging with the practice of scripting, revealing how both the verbal and the formal languages used by their members reveal changes in the way they learn about and conceptualize their practice as designers. Saliently, it highlights the tight coupling between students’ descriptions of their work – their discourse - and the technique employed. This coupling is described as constitutive of a metaphorical arena – an imaginary – in which the boundaries of machine, creativity, and intelligence, are re-qualified and re-negotiated through a computational “lens”. The article concludes by proposing the anthropological concept of depaysement and the notion of objects to design with as insights for casting the role of computation in architectural education and practice as more than just a tool.

1.1. Context

Representations as artifacts

The education of the architect seems distant, relative to other disciplines, from the engineering tradition of writing computer code. Inheriting from the Beaux-Arts tradition its meticulous care for the craft of drafting, and from the Bauhaus its disciplined intuition based on the sensual exploration of materiality and order, architects and architecture have come to epitomize a conception of creativity associated with the exploratory, the contemplative, and the intellectual, while computer programming is associated —at least in the popular imagination— with the production ethics of scientific capitalism. If sketching is construed as an open-ended, sensual, act, computer programming is construed as a highly structured process that requires pre-rationalization and a clear goal [1-4].

In contrast with sketching, there seems to be no room for ambiguity in scripting: computers lack intelligence—as well as practical knowledge—and a typographical error in a variable name, a call to a function, or in one of the language’s keywords, will cause the compiler to throw an error and prevent the script from running. The culture of programming is thus often thought to be in conflict with the more “free” or “unconstrained” nature of design, so often associated with the idea of creativity and invention. This distinction translates into the cultural archetypes of the geek and the creative artist, a binary that effectively builds disciplinary walls hard to debunk.
Recent research in social studies of technology has interrogated this binary and – more generally – the claims of objectivity that give authority to techno-scientific discourses by revealing the social, context-specific, collective and contingent nature of knowledge production. The “clean” nature of programming is put into question, for instance, by evidence that shows how technology design projects, such as the planning and programming of software systems, are decidedly contingent upon “messy” issues like verbal communication, hybrids of digital and manual techniques, as well as to political economies local to the development of the project [6].

In architecture, similarly, the process through which manual and verbal skills are learned takes place in the architecture studio. This is typically an open space where students develop “projects” through an iterative process of defining and redefining design descriptions such as models and drawings, guided by the critiques and reviews of their professors, who impart the language that architects use to refer to them, and encourage the good, or bad, practices they have acquired through their own schooling and practice.

Architects are thus expected to create compelling images and artifacts to excite society’s imagination. Depictions of possible futures, these images and objects become tools that allow different social actors to have a conversation about an unbuilt future. By referring to them as “design artifacts” instead of as “design representations”, the active role of representations as participants of such negotiations is highlighted. Design artifacts are not merely pointers, or descriptors; they are “objects to think with”. [7]

In what sense, then, can a script be thought of as a design representation or artifact? Scripting languages are, by definition, open-ended and it would thus seem impractical to attempt a comprehensive account of their possibilities. More generally it should be noted that scripts enjoy an ambiguous space between tools and descriptions. They are “performative” in more than one way: the outcome of scripts varies according to the code’s complexity and aim, but ranges from the simple automation of the placement of elements on the screen to the development of complex tools that interact with users, data, machines, or with geometries modeled “by hand”. The term “generative” helps describe their fluidity as design artifacts [8]. In appearance sketching and scripting embody fundamentally different “know-hows”, but a closer examination reveals their relationship to be much more interesting and complex.

Pocket sociology of the architecture studio review

Even in today’s PowerPoint age, an archetypical architecture studio review consists of an experienced professor, surrounded by his or her students, sketching on a piece of tracing paper (often one of the student’s drawings) while discussing specific aspects of the project under examination. Manual drawing remains still today a protected space for architects, and therefore
control over the pencil has a symbolic connotation of knowledge and power: as the pencil moves across the paper the students see how the person in command balances proportions, negotiates functional criteria, lays out perspectives, and even considers material qualities and technical feasibility in ways that verbal communication would simply fall short to describe. While the person speaks and draws, students tie form and action to discourse in particular ways. After the professor finishes his or her intervention the students incorporate these comments and sketches new drawings and models. As time passes new levels of complexity are added to the projects, and students gradually become acquainted with the language of architects and of architecture.

American sociologists Lave and Wenger have formulated a view of learning as a process in which participants gain increased access to expert performances through direct—yet gradual—involvement with a practice. This “performative” perspective on learning is developed in theoretical detail under the “peripheral participation framework” rubric, and has influenced studies of learning by accounting for the social and performative dimension of the “learning by doing” paradigm. Under this light, skills are not only reproduced through apprenticeship, but also changed, and change too the participants of the apprenticeship process. Architecture studios can be seen, under this light and within Lave and Wenger’s framework, as places where learners move gradually toward more intensive and engaged participation where participants advance from simple to complex projects by observing and gradually implementing the verbal and non-verbal practices of their instructors. In the architecture studio, control over the pencil and over the word during a review demarks a social order among the establishment a hierarchy where the instructor occupies the center and the

\[Figure 3.\text{A student discusses a script at the Bogotá Workshop, 2009 (student: Fernando Sierra).}\]
students stand at the periphery [9]. In the studio participants gradually gain legitimacy to undertake larger problems by engaging in increasingly complex projects and through the mastery of two kinds of tools: on one hand, the analytical and pragmatic tools for creating design artifacts (such as models, drawings, and 3D digital models), and on the other, their ability to explain their projects in an adequate “key”, that is, ciphered in the language of contemporary practice. Such is the intensely social and verbal aspects of design learning processes. How do scripts and computer programming enter this context?

2. Design Machines

2.1. Random density

The members of Team A have recently enrolled in the undergraduate architecture program at Universidad de Chile. They have had intense contact with CAD systems and their school, like virtually all architectural institutions today, offers CAD courses that focus on building drafting and modeling skills at basic and advanced levels. When using a drafting and modeling CAD system such as AutoCAD or Rhinoceros, they draw lines and polygons, repeat them, extrude volumes and twist them in operations that metaphorically mimic the drafting and the model-making table of the traditional designer.

When faced with the different ethos of scripting during the workshop, they are fascinated by the generative potential of code. At the beginning of the workshop, one of them said that writing scripts “Is like finding a new way of thinking, in a way it becomes more scientific, more rational. I don’t think I will design in the same way from now on.” The added layer of automation brought up challenged their traditional engagement with tools for architectural representation, letting them go under the surface of their modeling software, turning them into toolmakers, instead of just tool-users. (they in fact called their project “density machine”). For members of this team, this “deeper” engagement with the technology meant opportunities for experimentation, excitement, and discovery, and a new imagination of their identity as a more “rational” and “scientific” agents.

Team A decided to write a script for generating objects in an unpredictable way by implementing a simple algorithm that places volumes randomly in a three-dimensional matrix. Their idea was to subsequently constrain the unpredictability of the outcomes of the script by including certain parameters such as size and density. Their imagined role as designers is shaped by their decisions about what aspects of the models are randomly generated and what aspects are explicitly defined in the script—or through “user” input.

The automation of the “drafting” part of the design leaves an open question about the nature of the design act itself. This question reveals itself, tacitly at first, more explicitly in the end, in the language used by the team,
in which they saw the machine as “breeding” designs, and themselves as mere guides of the process by virtue of their ability to select from a variety of forms “generated stochastically in a semi-autonomous manner.”

The words and metaphors used by the team suggest a reconceptualization of both design and their role as designers: rather than drawing or modeling, they are choosing which outcomes of the process are valid or interesting design alternatives. Worthy of notice is a hybrid of evolutionary and computational terms and the playful attribution of agency to the computer. As toolmakers, members of Team A saw themselves differently from the designer they were used to enacting in their design studios. This enabled them to enjoy aspects of design and of computation that redefined both. As a side-result, a new role came into the equation. Repeatedly during the process members of Team A spoke in terms of a “user”. The “user” points at the black boxing of design functionality that of team’s description – scripts afford.

2.2. An automated draftsman for complex forms and fabrication

The members of Team B are two practicing architects and, at the time of the workshop, young faculty members at Universidad Peruana de Ciencias Aplicadas. They recently found it impossible to build an installation project using “traditional” CAD methods because of its geometric complexity, and thus decide, after the first workshop session, to use scripting to bridge the gap between digital descriptions and physical artifacts. Their idea is to develop a script that generates 3-Dimensional patterns in a non-deterministic way and prepares the surfaces for fabricating the components in a 3-axis numerically controlled machine. Voronoi patterns are often seen in nature in places such as, for example, arrays of soap bubbles, in the patterning of skin cells, or in the tiling of a turtle’s shell. In three dimensions, Voronoi patterns display irregular structural, as well as visual properties that
make them interesting to designers. The irregularity of these patterns pose a
design challenge as each tile of the Voronoi needs to be drawn individually
so that designing one of a certain size can take many hours of drafting and
modeling, and the end result would be a large, hard-to-edit digital file.

The members of Team B spend long hours trying to formulate the basic
structure of the script, one that enables them to automatically compute
different patterns. After many hours of revising the scripts, writing “pseudo-
codes” – a kind of sketch version of the script- and doing schematics, they
deide to simplify their task and focus on solving the fabrication of the
object. They start implementing a script that takes a cloud of points and a
surface modeled in Rhinoceros and they rationalize it into a set of smaller,
flat, numbered parts that can be milled out of plywood sheets using a
numerically controlled router. This way they hope to overcome the
complexity of fabricating non-planar surfaces from planar materials. The
script would also be useful, once they figure out the pattern algorithm, for
turning the large number of flat surfaces that compose it into an orderly
layout of labeled pieces ready for digital cutting and assembly.

Figure 5. Early results of the
flattening and labeling algorithm. 3D
Model by Natalija Bojlsakov and Brian
Miller.

When attempting to figure out the pattern, the team faced problems of
a different nature than those encountered when drafting or modeling. They
dealt with concepts such as iteration, encapsulation, and recursion, as well as
variables, functions and their scope, to automate the generation of the
geometry of their project. Even though they were not able to develop the
full script during the workshop, these concepts and words became their
tools for thinking about the problem. They thus experienced a different kind
of engagement with the practice of design. Having to figure out the
geometric process and mathematics behind the form gave them a rewarding
feeling of “going beyond” the visual. During the workshop, the team
switched back and forth between a divide-and-conquer strategy for solving
the geometric complexity of the project, and their own visual ideas about
the space.
In contrast with Team A’s project, the way design is defined implicitly in the script speaks of a conception of design in which the fabricating logic and its representation cannot be separated. This probably reflects their greater experience in the field both as students and as practitioners. For both teams, however, the layer of automation enabled by scripting became a vehicle for formal discovery and exploration.

![Figure 6. The ‘Scripting Voronoi’ installation, an exhibition space for student projects, designed and scripted by students of the scripting workshop Natalija Bojlsakov and Brian Miller.](image)

3. Conclusion

This article has assigned relevance and discussed emerging conceptions of design visible in the practices and language used by architecture students learning to script within CAD systems. The study shows how for the students of the workshops exposure to scripting meant also the appropriation of techno-scientific terms that informed the metaphors they used to conceptualize their designs. How to think about this reconceptualization as an ingredient of a rich architectural practice and pedagogy?

3.1. Depaysement

Anthropology’s notion of *depaysement* as the process through which “One leaves one’s own culture to face something unfamiliar, and upon returning home it has become strange – and can be seen with fresh eyes” [7]. This concept may be useful to think of the integration of computational theories and techniques into architectural design. The notion of *depaysement* (literally “to de-countrify”) speaks of the process of adventuring to new (conceptual, technical) territories, as a way of learning something new about our own
familiar domains. Designers who engage with computer programming through scripting are not only learning the language’s syntax and its control structures.

By learning computer programming architects also learn something new about architecture and about design; they put different epistemologies of design in dialogue and use new languages to articulate design problems. The members of Team A, for instance, discovered in the scripts not only a new way of generating form, but also a new way of describing design problems and strategies, and a new way of enacting their roles as designers. They have, in their imaginations, become more like toolmakers and machine builders, or even breeders of an “autonomous” process of architectural evolution. When they decide what to automate and what to leave as user input they are, perhaps inadvertently, making commitments about what design means to them.

For Team B, “iteration”, “encapsulation” and other computational terms became currency for describing spatial structures as well as computational constructs. As descriptors of spatial ideas these “verbal spaces” and metaphors became in themselves design tools or, as suggested above, artifacts. It also gave the members of the teams a deeper understanding of computers, and a feeling of empowerment as makers—rather than just users—of technology.

3.2. Objects to design with

In her study of computer culture in the United States Sherry Turkle used the expression “objects to think with” to refer to personal computers in relation to the individual and collective identities of their users and makers. The technological artifacts we create and use, she thought, spark our imagination and provide us with new ways to interpret the world. Similarly design artifacts, such as scripts, are not passive recipients of our intentionality – of our designs – but also condition the material products of our efforts as designers and our interpretive stance before design problems.

In line with social scientific perspectives on the construction of knowledge, the evidence presented in this article shows how scripting reveals the concept of design to be fluid and contextual in ways yet to be fully explored. Recognizing it as such may prove key in devising a richer integration of computational theories and techniques into the practice and pedagogy of architecture, one that recognizes the performative nature of architectural representations such as scripts and other parametric constructs without forgetting the performative quality of other “traditional” design media. In other words, a discussion about scripts in architectural pedagogy needs to stress their performative nature as representations while at the same time, through depaysement, help unveil the performative nature of any design representation.
Developing a critical sensibility towards the active role that technology – tools and methods – plays in design is crucial to architectural and design practice and education.

Acknowledgments

Thanks to my colleagues Kenfield Griffith and John Snavely, who planted the seed of the workshops that are the material of this paper, and all of our students in Chile, Colombia, Peru, and the US. Skylar Tibbits was a great co-instructor during the workshop taught at Bogotá. Special thanks are due to Pablo Herrera for the vision and coordination of the workshops in South America, and to Andres Gonzalez and Bob McNeel for valuable logistic support. I am indebted to Professors Heather Paxson and Terry Knight at MIT, and Lucy Suchman, for many insightful conversations around the subject of this article.

References


Daniel Cardoso
Massachusetts Institute of Technology
Computation Group, School of Architecture
77 Mass. Ave, Of. 9-269
Cambridge, MA 02139, USA
D. Cardoso, dcardoso@mit.edu