Software: Gradually and Then Suddenly

Daniel "Drex" Drexler
Center for Science, Technology and Society at Drexel University

Abstract

A study of software, the way it materializes perspectives, and the limits of the promulgations of those perspectives.

Software: Gradually and Then Suddenly

It's Software's World, We Just Live in it

Software worms its way into every nook and cranny of our world. Part of this paper is based on work I did *in* software, but the paper itself has also moved through software. Systems that were once realized physically have, for reasons beyond the scope of my work, moved into software. This means that academics use software to compose and share their ideas, while farmers fight companies for access to the software that runs their tractors (Koebler, 2017). Many of the roles that software has stepped into are not new. This project and paper are not founded on the idea that software allows anything that is unique per-se. I do believe, however, that software (and the artifacts that emerge out of the software process) have particular material qualities that change how they are in the world and the way they impact the world (Kitchin & Dodge, 2011; Mackenzie, 2006). This work began by trying to understand how the ideas of situated knowledges and strong objectivity can inform the creation of software (Haraway, 1988; Harding, 1992). Software, like all material things, encodes the perspectives and assumptions of its makers. But what use is it, really, to point out that the assumptions of Software makers remain in the things they produce? I hoped to find a path from finding and detailing perspectives encoded in software to a direct intervention at the software level (Zuiderent-Jerak, 2015). My approach assumed that details in the material qualities of software that would be useful in making a situated intervention into the perspectives encoded into it. Ultimately, it was the material aspects of how software exists differently for different stakeholders in its creation and use that would come to dominate that rest of the project.

I believe the limits to applying situated knowledges to software lie within the blackbox of software itself (Latour & Centre de Sociologie de L'Innovation Bruno Latour, 1999). Once the components that compose software are removed an examined, it becomes clear that software developers are always looking at the components inside the blackbox, while users always see it from the outside. The understand what must be changed involves working *across* black box layer(s) in a way that is unique to software. Though Latour's blackbox framework of encapsulated functionality is a useful starting point, the reality of software is that every artifact of the software process relies on action across many levels of blackbox'ing (Latour & Centre de Sociologie de L'Innovation Bruno Latour, 1999). It goes beyond Latour's framework and into a kind of object that I call a *composite*, an object that itself is a black box and which contains black boxes, but where Latour claims that blackbox boundaries become socially visible only in failure, *composites* are designed to use heterogeneous interactions across the boundaries of its constituent blackboxes. As with other blackboxes, the things inside a blackbox are normally invisible, and so the limits of situated knowledge are found in the difference between the situation of the user and the developer. The user, having no way to assess or understand how many different components are involved in each software component, can only communicate about the experience that emerges out of the final *composite* of all components. The developer, however, is always making changes to a particular blackbox at a particular level and, though they also use the software as a user, the path from that user-experience to understanding how to fix problems (or how the developer's standpoint has impacted a given software-artifact) is unclear. This quality of software comes from how software is assembled into composites and shapes how the perspectives of those who make software come to be encoded in software artifacts. The views of those who produce software are reliably represented, but those representations exist at different levels of the *composite* how each materialized perspective comes to be seen by users of the artifact is highly individual.

Returning to the question of what one can do to address perspectives in software, this project takes the Free Open Source Software (FOSS) project Emergency Development ENvironment (EDEN) as its object of study and site of intervention. I chose EDEN because it is, as far as I can tell, a well-made piece of software that fulfills its role in the world. Others will engage socially problematic software and try to fix it, but the goal of my work here is to engage with a piece of software which is not obviously harmed by the standpoints of its developers. This does not aim to be a critical engagement, but an engagement whose successes and failures may be informative for future critical projects. The result of the modifications to EDEN can be seen at <url>. The changes to EDEN highlight where EDEN crosses blackbox boundaries, though my efficacy in making those crossings apparent is limited to particular *kinds* of boundaries. As we will discuss, software is deeply heterogeneous and that heterogeneity impacts the work that is needed to engage with it.

What can we Say about Software?

The Social World of Software Makers

A great deal of excellent sociological work has been done on communities that produce various kinds of software. The communities that have been studied often have particular qualities of their social structures connected to their shared work of creating, maintaining and promoting software (Kelty, 2008). Other work has focused less on the work of making software per-se and has instead highlighted how ideas of freedom, creativity and openness (of a particular kind) attract people to communities that make software (Gabriella Coleman, 2012). These ethnographic engagements with communities that produce software share a perspective that certain kinds of social structures are better suited for producing software (in general) and *certain kinds of software in particular.* The FOSS movement, Kelty claims, has a social structure that flows directly from the ethics of sharing and attribution that unite the software projects under the FOSS banner (Kelty, 2008). While there are important lacuna in Kelty (2008)'s somewhat utopian descriptions of FOSS communities, especially for non-male and culturally heterogeneous people, it seems true that social structure and technical goals co-create and co-evolve with each other (Dean, 2010; Penny, 2013). Similar conclusions are reached by Boellstorff (2015) when looking at online cultures and Ensmenger (2012) in his study of the history of the profession of computer programming. Boellstorff (2015) vividly shows the validity of considering people spending in time in digital spaces *isolated from* any actual world[1]referents. He freely notes relationships between the accounts his interlocutors give of their actual world activities, but understands that it's still valid to engage with those representations directly without trying to prove any truth-claims about them. Ensmenger (2012)'s work is historical rather than ethnographic, but also supports the ways in which the material demands of making software created social structures that have endured until today. There are consistent links between the nature of the virtual and the social structure of the actual people who create digital products, but each of those qualities can be examined independently without needing to fully understand the nature of the links. Gabriella Coleman (2012)'s focus, in particular, on "hacking" cultures that exist outside of any particular culture of "software" or "code" points to these social structures fitting particularly well with software, but not proceeding directly from software per-se (Drexler, 2019). This is in line with Cox and McLean (2013)'s examination of the expressive qualities of "coding" (which may or may not be a synonym for "software") and how they relate to the long history of speech as well as how the particular material qualities of computers open up new avenues for expression. For this project, these works suggest that EDEN does have a particular culture around it, especially because EDEN's makers (the Sahana Foundation) places the project within the FOSS world of software development and the Information and Communication Technologies for Development (ICT4D) world. Studying this particular social situation would doubtless be productive work and continue the conversation around the social character of collectives that create software and try to promote FOSS ideas in particular historical, economic and legal regimes. However, this work also suggests that while the a particular Software-Object and the culture around and within that Software-Object co-create and drive each other's evolution, they also suggest that either can be considered in isolation at a particular moment in time. Indeed, Kelty (2008) and Gabriella Coleman (2012)'s observations both depend on connecting independently observed qualities of *culture* and *particular FOSS projects* in order to draw out the co-constitutive qualities. This project leaves whatever co-constitutive cultural links that exist between EDEN and the people and cultures that produce it to others, though I am quite sure interesting work exists there.

The Social World Software Helps Construct for Us

Even without a direct ethnographic engagement with the team working on EDEN (or its users), previous work on software and its effect on the world has been useful for guiding the focus of this project. It goes too far to say that the nature of socializing has been transformed by digital networked communication, but society (and social scientists) are engaged in an ongoing effort to sort out truly new ways of using media and new forms for existing modes of socialization. Much of this work has focused on modern, media-focused ways of keeping socially connected with people (Humphreys, 2018; Jurgenson, 2019; Tiidenberg, 2018). This work is not about making software, but speaks to the power *of* software to structure and channel social activities. Like the social structures that exist around the act of making software, living with software pushes us to act in particular ways. Humphreys (2018) both highlights how modern practices of using global digital networks to keep in touch with each other proceed directly from more localized historical practices, and how the way we represent ourselves in digital spaces reflect possible social realities that only exist in those spaces. Software places is in new social configurations: a neighbor I have never met could see my vacation photos without asking me, I bring my assumptions developed over a career of software development for US software startups to my review of an internationally focused (but originally Indonesian) system for disaster management. Though each digital connection has a particular character and the field is difficult to tantalize

[1] Boellstorff (2015) rejects the idea that events in digital spaces (like games or chat rooms) are any less "real" than things that exist in the physical world and thus rejects the common language of "real" and "virtual." He instead talks about "actual" (physical) spaces and "virtual" (within digital worlds) spaces. I will be describing the physical world as the "actual" world when appropriate and will talk about both virtual space (space that fully exist in computers such as Second Life and World of Warcraft) and digital space (spaces that are displayed though digital mediums, but are constituted of a mix of real and generated media [facebook, instagram]). It is all "real."

accurately, many of the connections that have become more common bridge greater social distances than previously possible. Dean (2010) highlights this quality of digital spaces when she speaks about the collapse of predictability in how others will see us. The accessibility of digital spaces means that we may be famous "online" but anonymous in person, or visa-versa.

Glossary

Software-Object A generic term that describes the entire process that leads to the production of a series (or a single) software artifact. This includes the economic and culutral position of the organizing project, the social organization of those who write the code for the piece of software, the social positions of the software artifacts' users, and the software artifacts themselves. This term is used in opposition to the common "software" because of the potential confusion between a "software project" (the organizing structure of a Software-Object), a "program" (a Software-Artifact), and other linguistically confusing ways of speaking about the multi-layered process that produces "software.". 5

Acronyms

EDEN Emergency Development ENvironment. 4, 5

FOSS Free Open Source Software. 4, 5

ICT4D Information and Communication Technologies for Development. 5

References

Boellstorff, T. (2015). *Coming of age in second life: An anthropologist explores the virtually human.* Princeton University Press.

Cox, G., & McLean, C. A. (2013). *Speaking code: Coding as aesthetic and political expression.* MIT Press.

Dean, J. (2010). *Blog theory: Feedback and capture in the circuits of drive.* Polity.

Drexler, D. (2019, March). *Hack the planet.* `https://medium.com/@aeturnum/hack-the-planet-aaa4abc23bc8`. Medium. (Accessed: 2020-4-18)

Ensmenger, N. L. (2012). *The computer boys take over: Computers, programmers, and the politics of technical expertise.* MIT Press.

Gabriella Coleman, E. (2012). *Coding freedom: The ethics and aesthetics of hacking.* Princeton University Press.

Haraway, D. (1988). Situated knowledges: The science question in feminism and the privilege of partial perspective. *Fem. Stud.*, *14*(3), 575–599.

Harding, S. (1992). Rethinking standpoint epistemology: What is“ strong objectivity?”. *Centen. Rev.*, *36*(3), 437–470.

Humphreys, L. (2018). *The qualified self: Social media and the accounting of everyday life.* MIT Press.

Jurgenson, N. (2019). *The social photo: On photography and social media.* Verso Books.

Kelty, C. M. (2008). *Two bits: The cultural significance of free software.* Duke University Press.

Kitchin, R., & Dodge, M. (2011). *Code/space: Software and everyday life.* MIT Press.

Koebler, J. (2017, March). *Why american farmers are hacking their tractors with ukrainian firmware.* `https://www.vice.com/en_us/article/xykkkd/why-american-farmers-are-hacking-their-tractors-with-ukrainian-firmware`. (Accessed: 2020-4-15)

Latour, B., & Centre de Sociologie de L'Innovation Bruno Latour. (1999). *Pandora's hope: Essays on the reality of science studies.* Harvard University Press.

Mackenzie, A. (2006). *Cutting code: Software and sociality* (S. Jones, Ed.). Peter Lang Publishing.

Penny, L. (2013). *Cybersexism: Sex, gender and power on the internet.* A&C Black.

Tiidenberg, K. (Ed.). (2018). *Selfies: Why we love (and hate) them.* Emerald Publishing Limited.

Zuiderent-Jerak, T. (2015). *Situated intervention: Sociological experiments in health care.* MIT Press.