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.

Many of the things software does (and the ways that software does them) are accurately seen as old wine in new bottles, but

This state of affairs can lead to a kind of flattening of software, where the real material qualities of the object itself are obscured and black-boxed by the social practice around the object (Latour & Centre de Sociologie de L'Innovation Bruno Latour, 1999)

References

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.

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.

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