Social Epistemology Review and Reply Collective, 2014
Vol. 3, No. 2, 22-23. http://wp.me/P1Bfg0-Y9

A More Robust Software Ecology?

Trevor Croker, Virginia Tech

For my first collective vision post, I will do as many have done in the past and talk about some of the main scholarly pursuits that interest me in social epistemology. Specifically, I want to refer to a previous conversation within Social Epistemology and the Collective regarding software epistemologies." In his article, "The Social Epistemologies of Software," David Berry persuades us to take the seriously software epistemes as "a useful means of uncovering the agency of software and code for producing these new knowledges." Software, it is argued, is increasingly a mediator between us and the physical world (Berry 2012, 381). In response to Berry's article, Nathan Johnson and Damien Smith Pfister suggest that perhaps it would be more useful to reverse the analogy. Instead of an "ecological approach to computation," we should focus on "computational ecologies," as a means of getting to the root of these new social knowledges. Johnson and Pfister, uncomfortable with the binaries Berry proposes, ask us to look at "local micropractices" and the "intentionalites of code."

Largely, I agree with all the authors' points, however, I want to respond briefly to the way that we deploy "ecology" when talking about large communication networks. As we start to adapt ecological metaphors to the practice of computing, I am tempted to broaden the lens we use to tackle these issues. From my own viewpoint, an ecological metaphor can fall flat if we do not include the broader environment that surrounds one aspect of computing. At the same time, I think we can potentially lose a great deal of tacit and practical knowledge if we get lost looking at too broad of a picture. With that said, I have two primary thoughts regarding how we can best understand the development of software, first at an infrastructural level and secondly on a personal one.

David Berry's decision to look at web bugs, beacons and trackers I think is a very productive move towards an ecological understanding of software. By focusing on these set of technologies, I think we start to pull together wide networks of actors involved in the production of software. However, I would push the focus even further. It is impossible to talk about bugs, beacons, or trackers without at least referring to the broader infrastructure that supports the development and propagation of these software programs. There should be, to some extent, a mention of the hardware, people and places that run these programs. If we are going to take the metaphor of ecology seriously, wouldn't it be useful to bring the physical ecology of the network into the fold? While I fully understand the increasing mediated experience that digital networks produce, I also understand that much of the maintenance of the Internet, for instance, occurs through face-to-face interactions between network engineers and other computer professionals. This topic is explored by the journalist Andrew Blum in his book Tubes.

Of course, actually trying to weave these disparate pieces together is a challenge that I regularly run into. Software ecologies, it seems to me, are difficult to pin down when we try to look at them holistically. Johnson and Pfister's suggestion to look at micropractices

is likely the most rigorous way of studying these ecologies, but I wonder whether there is a way to tackle some of the broader ways we talk about "software." Even in the case of bugs, beacons, and trackers, these are systems that rely on placing code into multiple systems across geography. These technologies do not function like traditional software (running locally). Perhaps software is not the right word?

My second point is more individual and it is something that I think we all could participate in. Berry, Johnson, and Pfister all point out that these software technologies are not simply given to users. Instead, through the use of these technologies, we become agents that modify the code itself. To a certain extent this sharing of knowledge has been present in many previous forms of technology. Still, digital technologies are ripe for the reproduction and sharing of knowledge through multiple user interactions. If we start to consider our own "software ecology," we might start to think of ourselves as participants in the development of these shared software knowledges. This, I think, is what many individuals in the open-source, creative commons, and "modding" communities have already articulated. While I do not think that autoethnographic accounts of our own technology use will get us towards a broader understanding of software ecologies, I think it might be a way of cracking open the nut.

Contact details: tcroker@vt.edu

References

Berry, David M. 2012. "The Social Epistemologies of Software." Social Epistemology 26 (3-4) 379-398.

Blum, Andrew. 2012. Tubes. Harper Collins: New York.

Johnson, Nathan R. and Damien Smith Pfister. 2013. "Ecologizing' Berry's Computational Ecology." Social Epistemology Review and Reply Collective. 2 (4) 7-9.