



Yield to Open Source: Software Production and the Construction of Openness at the Epistemological Intersection of Agency and Language.

Introduction: Openness at the Intersections

“Titstare is an app where you take photos of yourself staring at tits,” (Miller, 2014)

Imagine that you enter a labyrinth. You are nervous and excited; you have a vague idea of the direction of the exit, but you are not sure which direction to head. When you finally find someone to ask, the response to your question is “Read the fucking manual.” If you were still close enough to the exit, you might choose to leave. The answer would likely not compel you to keep exploring.

RTFM¹ is a common, yet very unhelpful, response when a newcomer asks an “obvious” question in a technical space. A recent variation on RTFM is LMGTFY (“Let me Google that for you”), but the same unwelcoming message underlies it: if you do not know how to find “easy” technical answers yourself, you do not deserve to get help².

The technical space we are focused on is open source software. Free/Libre Open Source Software (F/LOSS)³ is a category of software that has a few central characteristics – it is generally made by people who are working for free, it is distributed without cost, and its source code (effectively, the innards of the software) is available for anyone to

¹ Commonly said to new developers asking questions. When a person new to a technology joins an online space (most commonly a chat room) and asks a question, a few things happen: 1. The reputation of the new person is evaluated. This reputation includes gender, links to other well-known community members, etc. 2. The recentness??? of the question - did someone else just ask this? If so, the response is “Read the fucking scroll back.” And 3. The “obviousness” of the question.

² Please note that there is no cultural precedent for politely telling someone to RTFM. “Please go scan the docs and come back if you have questions” is a rare response.

³ Free/Libre Open Source Software is the full name. Many scholars and community members strongly identify with the Free and Libre terms, although “OSS” is a more common abbreviation. It is the abbreviation that Nafus uses, and so I will also use it throughout.

change and improve. The free software movement was started in the 1980s as a reaction against corporations controlling the means of knowledge production and a “moral response to capitalist economies” (Nafus, 2012, p. 669). It was a fundamental shift to insist that the knowledge we produce via code should be free as in speech and free as in beer.



But responses like “RTFM” signal is a clear divide – between the people who “know” (who have agency and control of language) and the people who are trying to know (who want agency and to learn to control language). In this paper, I am conceptualizing this divide as a specific (but movable) point in an abstract geography created with the contributions of three authors.



Both Carol Cohn's “Sex and Death in the Rational World of Defense Intellectuals” and Dawn Nafus's “Patches don't have gender': What is not open in open source software” describe cultural and disciplinary boundary work through manipulation of agency and legitimation of language. This agency-language interplay is demonstrated in our RTFM interaction above. It is the addition of Thomas Gieryn's “Boundary Work and the Demarcation of Science from Non-Science: Strains and Interests In Professional Ideologies of Scientists” that helps illustrate how boundaries are formed, destroyed and constantly negotiated.



Below, I argue that open source software lives at the conceptual geographic intersection of agency, language.


Gieryn's boundary work

“It was clear that this was a man's world. Women could come, but only if they followed dude rules. It was only cool if you could roll with the bros.” (Kornblum, 2013)

Boundary work is, quite simply, the act of drawing boundaries between things like

academic disciplines or political parties. Boundary work can be used to help define what “counts as” a particular area of work (boundaries as inclusion) and define what does not “measure up” to standards (boundaries as exclusion). Gieryn presents boundary work not just as in/out sorting mechanisms but as ideological battles in which the weapons that both sides use are ever-evolving and context-sensitive.

Gieryn’s boundary-work is central not only for its lucid contribution to how and why ideological boundaries are formed, but because it serves two other purposes here. Firstly, it gives us a conceptual geography for the constructed boundaries between concepts and secondly, it provides an initial look at the geographical intersection of this analysis – the space where agency and language overlap. Because of this centrality, we will explore his conception of boundary work in depth.

Boundary work has existed for as long as there have been disciplines. In Victorian England, John Tyndall helped to draw boundaries around science to distinguish it from religion and mechanics. According to Tyndall, the aspects of science that make it scientific include its empirical nature, skepticism, objectivity, and practical usefulness. In contrast to religion, says Tyndall, science is *useful* (Gieryn, 1983, p. 785) In a different context, Tyndall described an entirely different picture of science’s intrinsic science-ness: science is knowledge that is grounded in experimentation, valuing discovery for its own sake. Science doesn't have to worry about being *useful*, he says, it has nobler concerns (Gieryn, 1983, p. 787). Here we see science, as a concept and ideology, being granted the agency to choose its role and its concerns. 

Tyndall's contradictory descriptions of science and scientific knowledge give great weight to Gieryn's argument that even as we try to draw boundaries around

"science" there is no objective science to which we can point. Science itself is conceptualized as a monolithic ideological concept that we can skew and manipulate for our ideological and rhetorical (and political, and strategic...) goals. We will see later how useful concepts are as weapons and tools of tyranny.

The uses and results of boundary work are not only ideological. Part of boundary making involves creating a technically-knowledgeable elite separate from the general population. This separation means that only those with adequate training can evaluate technical claims - there is no need to poll an uneducated populace to see if a claim is scientifically valid. By being a scientist doing work in a boundaried area, the scientist's claim is given respect and is thus unchallengeable by non-scientists. As a result, scientific knowledge makers are insulated from public interrogation and from the impacts and implications of their knowledge (Gieryn, 1983). This insulation furthers the divide between expert and non-expert language and is the foundation for the legitimation of language that is a key part of boundary work.

There are many ways that boundary work can be made visible, tangible or embodied and many facets of our conceptual geographic intersections. We will see how the overlap of two distinct actions—the transmogrification of agency and the legitimation of language—create a unique geography that is a natural location for open source technology production.

The manipulation and transformation of agency



There are three actors involved—objects, concepts and people—and each of them is granted a distinct form of agency. As used throughout, agency is used to mean the a state of having power or control, or a state of instrumentality.

Concepts with Agency



First, we give agency to concepts. Gieryn describes the way that science carries its own intellectual authority – science is valid because it is science. In all of our conversations about the future of technology, or even correlations between technology and forms of societal progress, we see technology as a concept being given agency and power (Mezzofiore, 2017). In Nafus, technology is an agent-concept it “[has] its own moral imperative” (Nafus, 2012, p. 676). Most specifically, in open source contexts, we give agency to the concept of openness. In F/LOSS spaces, the concept of “openness” carries with it a mountain of ideological baggage. There are certain behaviors and perspectives that are crucial to openness and violating them becomes violating the sanctity of openness itself.



As Nafus says, “openness relies on a steadfastly closed epistemological frame” (Nafus, 2012, p. 681) and this is true of any context in which a new boundary is being defined. In order to maintain the boundary, in this case between “open source” and “everything else,” ideological purity is required so that people can determine what side of the boundary they want to be on. Openness is used as a decider of what and who is valid.



Objects with Agency

Second, we give agency to objects. The defense intellectuals discuss bombs interacting with each other as though they were not launched by humans (Cohn, 1987, p. 697). In Nafus, code has agency. (Nafus, 2012, p. 676). Discussing projects, engineers talk about how “code runs,” as though the code itself can choose to do so and they “grant code its own agency” (Nafus, 2012, p. 678) . In F/LOSS contexts, programmers⁴ self-

⁴ I use the term programmers, coders, and software engineers interchangeably

identity with their ability to “write code that runs”⁵ and set themselves apart from the individuals who do other important software work like QA⁶ or UX⁷.

People with Agency / People without Agency

Third, and most significantly, we take agency from people and give it to the “experts” as Gieryn conceptualized them.

The defense intellectuals also use language as a manifestation of agency/positionality when “the speakers’ technostrategic language are positionally allowed, even forced, to escape” the idea of themselves as victims of nuclear war (Cohn, 1987, p. 706). The defense intellectuals remove all agency from citizens of countries that would be the subject of bombs, and give it to themselves as controllers of the future of

Code produces “transformative knowledge of self” (Nafus, 2012, p. 679) in the same way that technostrategic language use produces a great sense of agency. All speakers of the defense language share a transformed sense of agency and their detached positionality as the only ones who can control the bombs. Similarly, F/LOSS programmers are unified by their control of code and ability to produce technological artifacts.

In this way, there is boundary organization of programmers with a distorted perception of their power and importance. Because their code “runs,” because they are the creators of object-agents, they often conceptualize themselves as having a greater sense of control and centrality. Their boundary space, as Gieryn theorized, separates them from the general public and insulates them from non-technical interrogators.

Programmers, insulated from the users, the implications, and the long-term impacts of

⁵ This is common terminology in technology

⁶ Quality Assurance

⁷ User Experience Design – the study of the best way to present screen interfaces to end users.

their work, are free to maintain their technological elitism and superiority because of this culture. From a societal standpoint, non-technical individuals are being controlled (loss of agency) by the technology they need; programmers, however, are situated as being in control of it (possession of agency).

Another way that agency manipulation is seen in technology contexts is the obfuscation of agency of individuals with their jobs. In Nafus, the boundaries between which workers performed development tasks and which workers performed “support”

tasks like documentation or QA⁸ was divided almost perfectly along gender lines (Nafus, 2012). However, the women were hesitant to “acknowledge the materially obvious ways in which their participation had been socially shaped” (Nafus, 2012, p. 676). What we see here is an example of the women wanting to retain agency over their participation and their roles in the technological space (another example of the manipulation of agency common in F/LOSS contexts). It is disempowering to believe that you chose to do QA work because you were socially funneled into that area⁹.

The positionality resulting from the illusion of control and manipulation of agency reinforces the boundary between those with and those without control of technology and technological artifacts.

The legitimization of language

The other aspect to this conceptual intersection is the legitimization of language. There are three ways we can see this happening: by narrowing the kinds of responses that are acceptable to a certain question, by specifying the kinds of language that can be

⁸Quality Assurance. Typically code moves from development (where it is created by programmers) to QA.

QA work is feminized as “support” work in which women check the men's technological production.

⁹ A lot of work is being done, systemically, to give programmers (especially those just starting out) more agency and choice in their careers.

contained in those responses, and by overloading meanings onto words.

To illustrate the first, Cohn describes the ways that defense “discourse has become virtually the only legitimate form of response to the question of how to achieve security” (Cohn, 1987, p. 712). Cohn discusses how using common language to respond to a defense topic was unacceptable and the defense intellectuals would not consider her an equal in conversation. In F/LOSS communities, the de-facto (read: legitimate) way to get the code you wrote incorporated¹⁰ into the main project is to advocate for it in an issue queue flame war. This may sound like a dated “Hacker” movie concept, but is still a very real and very prevalent method of communication about technical matters. Nafus argues that language is also used to draw boundaries between those who “can take it” (which is also a form of masculinized boundary) and those who cannot. This is inherent in Cohn’s analysis of defense language - if you cannot handle the sexual overtones, then you are not serious enough to be a part of the conversation. Similarly, if you cannot handle the issue queue flame wars, you are not serious enough to be an open source programmer. This is a legitimization of the types of responses that are reinforced.

The second form that the legitimization of language takes is the regulation of the types of language that should be contained in a response. As Cohn finds that she is required to use traditional defense discourse as her type of response, she also finds that she is required to use specific defense words and phrases. For example, she had to use “escalation dominance” rather than an layperson version like “controlling the pace of the battle” (Cohn, 1987, p. 708).¹¹

¹⁰ Also known as “merging.” To “merge” code into the main project is to combine work in a separate location

¹¹ I made this layperson version up. A google search for alternative phrases for “escalation dominance” revealed that the term is so prevalent that it’s difficult to even find a definition. The closest I found was

There is an at least one side effect to this legitimation. Becoming subject to the “tyranny of concepts” means that “language shapes your categories of thought” (Cohn, 1987, p. 714). Concepts that were previously charged become neutralized by one’s participation in a context. This is an example of language as weapon (or language as *thief* of individual agency) – the defense intellectuals’ culture was so immersive and the language so powerful that Cohn found herself unable to access her own concepts of peace and justice after her enculturation (Cohn, 1987, p. 706).

The third legitimation of language is what I am calling “overloading of meaning.” In programming, engineers have the ability to “overload” a function¹². This means that you can have two functions with the same name but different contents, producing different results. Similarly, overloading a word is the ability to give two or more meanings to the same word, typically meanings beyond those one would find in a dictionary.

An example of overloading of meaning is the technological use of the terms “master” and “slave.” In computing terminology, you implement a master/slave database configuration most often for performance or security reasons. The “master” is the primary source for information, and can have an infinite number of “slaves” that are available for servers to be read from or written to¹³. Technologists use master/slave terminology readily, and without thinking about the origin of the words or their connotation.¹⁴ Companies (Reuters, 2003) have requested that terminology be changed,

a website defining it as “dominating the speed of escalation.”

¹² A function is a named set of instructions to do a specific task. For example, a function named ‘sum’ might add two numbers together and return the result.

¹³ See Figure 1 for a master/slave diagram.

¹⁴ A common use would be in a sentence like “If there’s too much traffic, we’ll just add slaves as needed.”



and open source communities continue to debate whether their “freedom” to create software extends to their “freedom” to use master/slave (Drupal.org, 2014)¹⁵.

Defense intellectuals also overload meaning. In their case, they use innocuous words as stand-ins for weapons terminology: “crew members call the part of the submarine where the missiles are lined up in their silos ready for launching ‘the Christmas tree farm.’ What could be more bucolic-farms, silos, Christmas trees?” (Cohn, 1987, p. 698). The intellectuals have normalized the use of familial and friendly words in serious and global contexts.

Note that while Cohn’s defense intellectuals describe violent technology with innocent abstractions, technologists describe (relatively) innocent technology with historically violent terms. This is a manifestation of the same mechanic – demonstrating group culture through idiosyncratic use of language that would be nonsensical in another context

In this way, Gieryn’s “tyranny of the concepts” is the center of our intersection. The concepts become legitimized through language and the scope of acceptable concepts is narrowed. The legitimized concepts in turn become agent-concepts as they are used as weapons. These agent-concept-weapons separate actors from the repercussion of their actions; they separate people from their thoughts; and despite being at the center, they also work to shape geographies and to form the boundaries we have been discussing.

Conclusion: Practical Implications

One societal perspectives that is relevant to our discussion here is the belief that

¹⁵ In many projects, the master/slave language has been replaced with “primary/replicant” and this has the advantage of also being technically correct.

“the technological is [...] to be orthogonal to the social” (Nafus, 2012, p. 674). This manifests in many ways, including contributing to the stereotype of the programmer as a solitary individual, creating code alone in a basement.¹⁶ One of its more toxic real-world manifestations is in the notion of technology being a space that is gender and race blind.



Coders believe that gender ought to be irrelevant (Nafus, 2012, p. 673) because the agency and focus belongs on the code and how it runs, not on the aspects of the creators of the code. It is a not-uncommon situation to hear men exclaim that women are sexist for bringing up gender, that people of color are racist for bringing up race when “I just want to write code.” This real-world transfer of agency – from humans to code – is central to many of the problems that exist in technological spaces today.



I believe that thinking about agency and language as manifestations of conceptual geographic boundaries can help draw them out of background activity - bringing attention to the actions that many participants take for granted as how things are done. There has been a sea change in F/LOSS communities in the last several years. Communities have introduced Codes of Conduct (CoC) for online spaces and in-person events.

Communities have created teams to deal with CoC violations and procedures for handling situations of sexual and behavioral misconduct. The news is full of prominent men in tech finally being recognized for their cultural mis-contributions to the culture of their organizations.

Conclusion: Final Thoughts

The objects we create (including the code that we write) are not value-neutral, and are not without context¹⁷. As a result, the technological is not orthogonal to the social,

¹⁶ See Figure 2

¹⁷ I could prove this if I had another paper to reference.

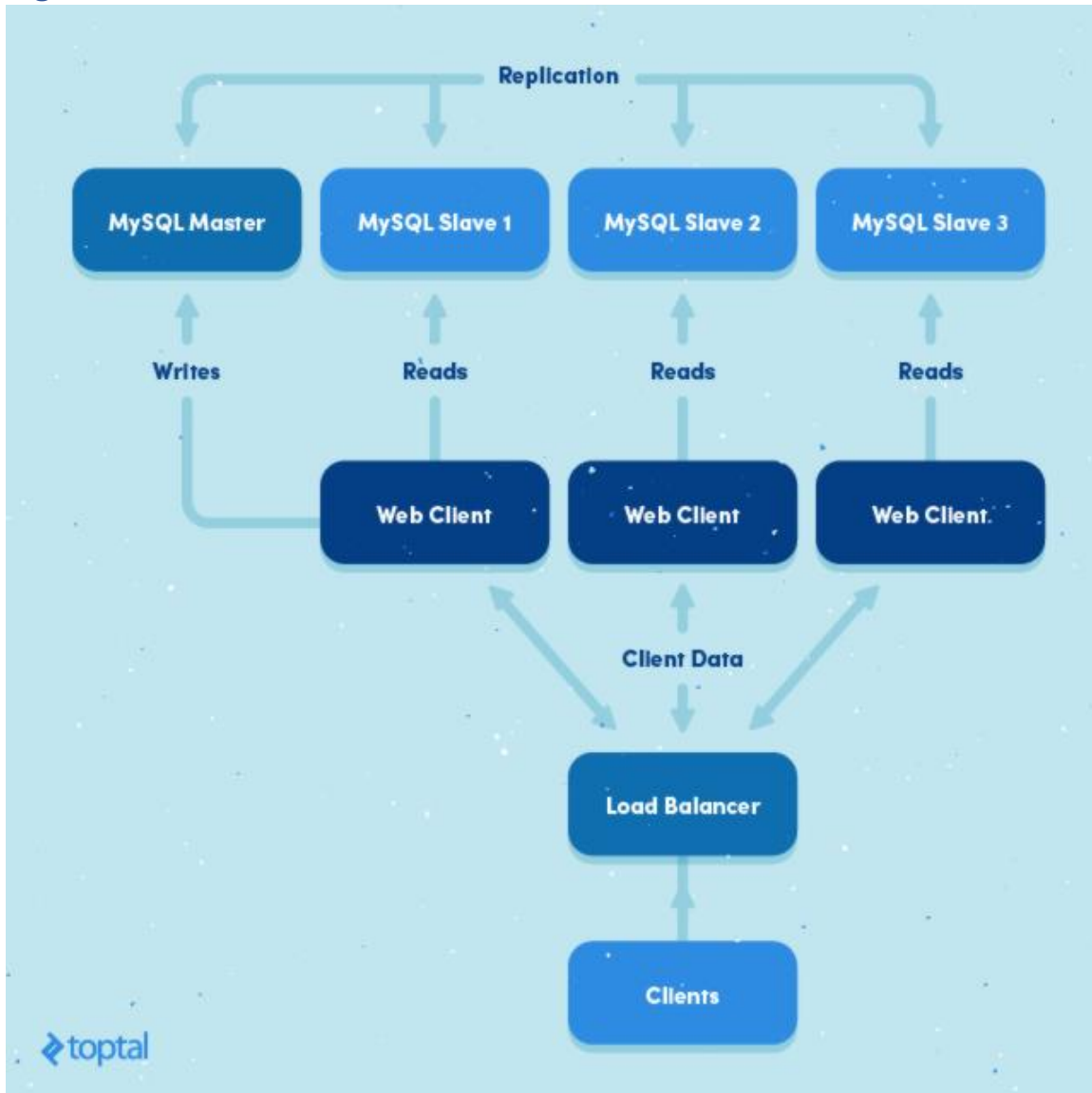
but they are co-constructed and with each new programming language or software project, or new iterations of existing products. Each interaction creates a new intersection and a new battleground on which the concept of openness and its constituent ideology can become a weapon against individuals who prefer different methods of technology construction.

Rather than conceptualizing boundaries as fixed, and the culture within them as static, Gieryn gives us a way to think of boundaries as fluid and strategic. When we are consciously aware of boundary-making activities (as Gieryn asserts we are), we can alter them as our needs evolve. These conceptual boundaries, like open source software itself, are not controlled by any one entity, but collaboratively made, reshaped, destroyed as agents overlap, intersect and enmesh.

Bibliography

- Cohn, C. (1987). Sex and Death in the Rational World of Defense Intellectuals. *Signs*, 12(4), 687–718.
- Drupal.org. (2014, May 28). Replace “master/slave” terminology with “primary/replica.” Retrieved November 10, 2017, from <https://www.drupal.org/node/2275877>
- Gieryn, T. (1983). Boundary-Work and the Demarcation of Science from Non-Science: Strains and Interests in Professional Ideologies of Scientists. *American Sociological Review*, 48(6), 781–795.
- Kornblum, J. (2013, September 14). It’s time to eliminate bro-culture from the tech industry. Retrieved November 10, 2017, from https://www.salon.com/2013/09/14/how_bro_culture_has_been_hardwired_to_the_tech_industry_partner/
- Mezzofiore, G. (2017, March 7). Stephen Hawking reminds us technology will kill us all and it’s all our fault. Retrieved December 8, 2017, from <http://mashable.com/2017/03/07/stephen-hawking-technology-kill-us-all/>
- Miller, C. C. (2014, April 5). Technology’s Man Problem. *The New York Times*. Retrieved from <https://www.nytimes.com/2014/04/06/technology/technologys-man-problem.html>
- Nafus, D. (2012). “Patches don’t have gender”: What is not open in open source software. *New Media & Society*, 14(4), 669–683. <https://doi.org/10.1177/1461444811422887>
- Reuters. (2003, November 26). CNN.com - “Master” and “slave” computer labels unacceptable, officials say. Retrieved November 10, 2017, from <http://www.cnn.com/2003/TECH/ptech/11/26/master.term.reut/>

Figure 1



<https://uploads.toptal.io/blog/image/91937/toptal-blog-image-1452523127725-dddccc82ac412d0d1d8dd2cb33c5f7084.jpg>

Figure 2



<http://theshelternetwork.com/wp-content/uploads/2014/08/south-park.jpg>