Adrian McKenzie

**Full-stack infrastructures: combinatorial delirium in code repositories**

Adrian Mackenzie, Sociology Department, Lancaster University.

a.mackenzie@lancaster.ac.uk

Recently released NSA documents revealed the existence of FOXSCORE, a massive database that bugs traffic from major internet exchanges. General Keith Alexander, Director of the National Security Agency, assured the public that the program is rubber stamped by a secret court. (http://divergentdave.github.io/nsa-o-matic/)

Recently released NSA documents revealed the existence of WAGONWIND, a USB hardware host tap that deanonymizes communications satellites. An anonymous administration source assured the public that the program would only be used in the event of a national emergency. (<http://divergentdave.github.io/nsa-o-matic/>)

The NSA-o-matic is a very small piece of code that generates **names** of potential US NSA (National Security Agency) information infrastructures. We know from the files released by Edward Snowden something of the existence of sprawling cyber-security infrastructures such as XKEY, TREASUREMAP, and so forth (CJFE 2015). One of the problems in making sense of contemporary communication infrastructures is the sheer variety of channels, modalities, platforms, scales of operation and levels of infrastructural interdependency. The Snowden revelations, in all their confusing variety, attest to a kind of combinatorial delirium in generating new ways of interacting with infrastructures. The NSA-O-Matic plays on this combinatorial process to generate a litany of possible NSA infrastructures. The humour of the NSA-O-Matic derives partly from the disparity between the existence of seemingly senseless juxtapositions -- USB hardware that deanonymizes communications satellites? -- and our still somewhat inchoate sense of infrastructures growing tendrils into every aspect of contemporary communications (wireless routers, skype conversations, network hubs, mobile phone conversations, social media, email messages, etc.) against a threadbare democratic mandate for their existence. The public assurances given by government figures such as General Keith Alexander or Senator Dianne Feinstein contrasts with the perfunctory and weak oversight. The hard infrastructure of communication and the soft infrastructure of justice or government seem radically misaligned.

This combinatorial delirium is not unique to the NSA or government intelligence agencies of a similar ilk. Indeed, their assiduous and passionate commitment to infrastructural access is well and truly part of the contemporary combinatorial processes I would like to focus on here. NSA-O-Matic points to a more general situation or to a process that elicits many different responses, on various scales across diverse cultural and social spheres. There is no single logic to the combinatorial exuberance at work in networked communication infrastructures, since it is the product of multiple determinations. We might look at the sheer abundance of 'apps,' relatively small pieces of software that connect mobile devices like phones into global-scale computing and communication infrastructures as another site of the combinatorial exuberance.

What I like about the putative infrastructures that NSA-O-Matic spoofs is that it relies almost completely on names and the evocation of infrastructures by naming. While strenuous and important efforts are being made to map the connections between the NSA's 'programs' and the communications and computing infrastructures that many of us use daily, it might be worth staying close to the naming practices that generate the combinations that NSA-O-Matic displays. My suggestion here is quite simple: we might think about a certain proliferating, unfurling and unstable aspect of contemporary infrastructures by examining only the names of pieces of code repositories. I tend to assume that we have already heard almost to the point where we cannot hear it anymore the claim that infrastructures are made, managed, maintained, shaped and understood in terms of software and code: 'infrastructure is code.' Code and coding have been widely discussed as making contemporary infrastructure what it is. The conceptual genealogies I draw on include science and technology studies accounts of information and scientific data infrastructures (G. C. Bowker 2005; Edwards et al. 2011), as well as anthropological accounts of software in terms of recursion and freedom (Kelty 2008; **???**) (although it lies quite a long way from debates about open vs closed source code, the debate that monopolised attention to software during much of the last decade) and connections between network infrastructure and contemporary urban experience (Graham and Marvin 2001; Thrift 2014).

Like NSA-O-Matic, I propose to remain very superficial in my description of the relation between coding and infrastructures. This runs somewhat counter to tendencies to see infrastructure as the underpinning or hidden truth, the static global variable (to use coding talk) of contemporary experience and its economies. Rather than analysing infrastructures in their obscure depths, it might be worth engaging with what could be called the *nominalistic semiosis* of code-affected infrastructures.1 Therefore, I approach software less in terms of digital media platforms or protocological control systems, and more in terms of a permeable network of names threaded through many different media, communication, administrative and operational infrastructures. Naming practices offer a viable way of navigating the densely trafficked archipelagos of code, protocols, standards, devices, diagrams, statements and operations. Methodologically speaking, focusing on names might seem a bit counter-revolutionary, since attention to practice, materiality, devices, things, non-human actors and generally extra-discursive elements has been one of the hallmarks of the attention. Given that naming, however, is a practice that links people and things, places and devices together in complicated associations.

***Github as infrastructural semiosis?***

It is quite difficult for social researchers to study these combinatorial practices directly, but there is one place where the combinatorial play can be mapped: online code repositories. The specific infrastructure I plan to discuss is the largest repository of code, the code repository platform (Github.com)[http://www.github.com] with its millions of software repositories sprawling across many domains. Code repositories store the source code, associated documents, files, configuration information and a variety of other materials associated with software.2 Github is rather like NSA-O-Matic writ large, since the millions of code repositories it hosts not only address almost any kind of digital hardware and software one can imagine, but seem to be animated in their growth since 2007 by the same combinatorial delirium that NSA-O-Matic, itself a Github-hosted project, plays with. It is difficult to characterise the heterogeneity of things, devices, conventions, platforms and organisations present on Github itself, and indeed, this difficulty is one of the motivations to describe what happens there in terms of names.

In some ways, Github looks like a typical contemporary social media platform since it has much of the social media appurtenances of participation (followers, groups, watchers, stars, etc.; see (John 2013) on participation) and invokes the ethico-economic injunction to 'share.' But what Github seeks to socialise is coding as a practice, a practice that takes place in many domains and across boundaries between devices, infrastructures, institutions and things. Github epitomises contemporary attempts to 'socialise' digital devices, infrastructures and networks by rendering them all as code-like entities, but at the same time, it treats coding as a social practice of participation, somehow akin to photo-sharing or chatting with friends using a social media platform. On the one hand, it is a social media platform on which infrastructures become the contents of social rituals and enactments. In this respect, Github might be seen as a meta-infrastructure. On the other hand, the trappings of social media and its imperatives -- 'be social, share code' -- with their focus on networks of relations between people largely occlude the tremendous combinatorial processes occurring on Github and git-like platforms more generally. The fluxes of imitation of names and their recombinant energy constantly overflow and elude the constraints of the imperative to 'be social.' Hence, Github is always already a somewhat 'splintered' meta-infrastructure (to borrow Graham and Marvin's still resonant term).

What access to infrastructure might be afforded by names? It turns out, or least this is what I seek to argue, that names and naming offer a kind of grounding in the relational adhesives of contemporary infrastructures. If we viewed, as we cannot, the 13 million or so unique code repositories hosted on Github in 2015, we would first encounter a rather complex baffling array of repository names. Repository names on Github can be any combination of letters or characters. These range from the almost casually random, yet still recognisable repositories named asdf (the first four characters on the home-row of an English keyboard) or 1234, both of which number in their thousands through to highly recognisable names such as linux or apache. Both of the latter projects are well-known as key infrastructural elements of contemporary digital networks, the linux kernel in particular representing one of the main ways in which coding work has become more visible as a cultural and social practice (Coleman 2012; Kelty 2008; Mackenzie 2006), and practically permeates the information infrastructural *stack*. The stack comprises a diverse range of materials that could include storage systems (disk drives), servers, network resources, hardware specificities, data models and database architectures, business and transactional logics, software frameworks that connect elements of systems and architectures together, user interfaces including how screen elements such as graphics, interactive devices and text operate together. The first four layers of this stack, for instance, would be dominated by linux kernel versions, since the Android operating system, which has the largest installed base, is a version of linux. Or, if we turned from handheld devices to supercomputers, 97% reportedly use linux. This is clearly a repository of no little significance. While apache is perhaps less well-known, it has for over a decade now has served the largest share of websites (around 40% or 340 million sites in January 2015 (Netcraft 2015)).

Between the two extremes of the random keypress repositories and the installed bulk of linux or apache stand a very diverse and densely populated range of repositories whose names could, if read serially and in parallel, tell us quite a lot about what how infrastructures take shape, multiply, combine, and decay in full-stack. Processes of repair, re-invention, appropriation, capitalisation and speculation might start to take on a more fine-grained consistency here, but most importantly, we can already begin to see that infrastructures, insofar as they are articulated in code, rely on a *nominalistic semiosis*. The gamut of millions of names of Github repositories, enough to populate a small country, is somewhat heterogeneous, pluralistic and full of transient multiplicities, but axes of organisation that assemble infrastructural elements into something like the 'full stack' associated with the digital networks can be discerned there. The distance between repositories called 'asdf' and 'linux' is hard to imagine because the former typically represent ephemeral contacts between people and infrastructures (for instance, many such repositories are the traces of people trying out or learning to use Github) and the latter encompass whole network architectonics. Movement along the axis between ephemeral and durable (the linux repository has been hosted on Github since 20073) is relatively easy on Github itself since that platform, with its infrastructures for storage, searching, tracking and graphing of coding work, arrays all repositories on the same surface. In this respect, it renders all infrastructures superficial, and this superficiality is perhaps a somewhat novel experience.

***The namespace -- a way of thinking about infrastructural couplings***

Infrastructures like Github induce the nominalistic semiosis of stacked infrastructures more generally by allocating all things a position in a *namespace*. Nearly all software code heavily relies on naming practices. Whether viewed as a series of statements, commands or functions, naming and the forms of address associated with names weave inextricably through software. It is only more recently, however, that the relations and connections between disparate bodies of code of many different ilk have all started to take place in a common namespace. Naming practices themselves, moreover, display specificities that might help us analyse the differences and associations between people and infrastructures are re-iteratively re-drawn. For instance, on Github, the names of code repositories when fully specified have a number of components. Take the 70,000 or so repositories whose names include the term JQuery. JQuery is a software library that web developers use to design the kinds of interactive webpages full of buttons, animations, dynamics menus, etc. that we associate with say browsing the web or buying a train ticket online. The details of JQuery need not concern us here, but it is interesting that its popularity amongst web developers -- more than 50% of all websites were making use of it in 2012 (**???**) -- arises from the way that it allows webpages written in HyperTextMarkupLanguage (HTML) to be manipulated as a set of named elements. In other words, JQuery subjects the display screen as a set of named elements subject to operations. Statements written using JQuery operate to generate text and graphics that people read, scroll, click, and select. This relatively banal example of how a page becomes a field of named entities pertains much more generally. If for instance we turn to the much more obviously infrastructural operations of something like Amazon Web Services Elastic Compute 2 (AWS EC2), something similar appears. The approximately 21,000 Github repositories relating to AWS EC2 display a great range of concerns, but nearly all offer ways of naming and addressing instances of linux services running in Amazon's data centres. These may be implemented in different programming languages -- Haskell, Ruby, Python, Java or Javascript -- and many different naming conventions permeate these different languages, but all of them rely subject the operation of elements in AWS EC2 to invocations.

There are 70,000 JQuery repositories or 20,000 AWS EC2 repositories. How are they related or distinct from each other? The fully qualified naming of things is an important pre-occupation in contemporary infrastructural work. In the case of Github, every repository has a unique name, even if it is a direct clone of another repository. The basic form of this qualification combines a person or organisation and a thing, device, system, convention, protocol or infrastructure. The base name for all the copies of JQuery is jquery/JQuery. The left hand part of the name is a person or an organisation, in this case, the JQuery organisation or the [jQuery Foundation](https://jquery.org/), while the right hand part points to the code itself, code that will inevitably contain many further named elements that need not concern us here. The name of a repository always combines a person or organisation (BBC, Google, Twitter, WhiteHouse, torvalds, etc.) and a thing (jQuery, bootstrap. linux, dotfile, etc.). Many repositories differ from each other *only* in name. That is, some thousands of the jQuery repositories will be strictly speaking *clones* of the primary jQuery/jQuery (or the jQuery Foundation's primary jQuery repository) differentiated only by the person or organisation with whom they are concatenated. Thus the fully qualified name of repository inextricably points in more than one direction. It does not index or symbolise either people or things but always a concatenation of organisation/person/device/thing.

Such concatenations are familiar in science and technology studies accounts of technologies and scientific knowledges (since they, for instance, lie at the base of actor-network theory accounts of translation and displacement (Callon et al. 1983) or human-machine configurations (Suchman 2006)). They do, however, intensify in contemporary infrastructures because the very operation of the thing/device/system/framework/protocol/tool/technique and the position assigned to the subjects who make statements about the infrastructure come much closer together and indeed mingle in code. Strictly speaking, the subject position marked by the left hand end of a Github repository name overflows the / that separates it from the body of code that addresses devices or systems on the right hand side. The situation here is captured well in Michel Foucault's account of the constituting and generating operations of statements in *The Archaeology of Knowledge*:

In each case the position of the subject is linked to the existence of an operation that is both determined and present; in each case, the subject of the statement is also the subject of the operation (he who establishes the definition of a straight line is also he who states it; he who posits the existence of a finite series is also, and at the same time, he who states it); and in each case, the subject links, by means of this operation and the statement in which it is embodied , his future statements and operations (as an enunciating subject, he accepts this statement as his own law) (Foucault 1972, 94–95).

Repositories comprise sequences of statements instituting the operating limits, the patterns of inscription and transcription, the regularities and usages that comprise infrastructures in practice. These statements take effect, or implement something only through the coincidence that Foucault describes here. The subject of the statement is also the subject of the operation, Foucault claims (and he is describing here what he terms a 'discursive formation'). This is reminiscent of more recent accounts of performativity and things, but the embodiment of the operation in the statement -- and effectively I am suggesting that the repository name refers to such statements -- also mobilises future statements and operations. How so, in terms of what happens on Github repositories?

One way to discriminate this linking of subjects and operations appears in all the repository names associated with jQuery that are not simply clones of the jquery/jquery repository. That is, the vast majority of the repository names in which jquery appears also contain some other name. Composite names comprise much of the proliferating bulk on Github at various levels. For instance, names such as:

* jquery-postcode-anywhere
* jquery-bingsearch
* select2-json-php-mysql-bootstrap-jquery
* jquery-django-formset
* twitter-jquery-plugin

display a range of infrastructural compositions occurring around jquery, ranging from postal address systems (postcodes) through web search engines, layers in the internet stack (web servers, databases, web frameworks) and social media platforms. This sample of composite names is by no means representative since much of the compositional semiosis around jquery pivots on individual elements of webpage user interfaces such as forms, menus, scrollbars and so on. Whether they stick closely to the graphic display of text and image or range more widely into the linkages between infrastructural layers, the combinatory composition that begins to appear in these repository names operates much more widely in mobilising infrastructures. Very easily, I suggest, we could find imitative and combinatory fluxes running across the millions of repositories on Github just at this banal level of names and naming. Other imitative and combinatory fluxes run much more deeply through the elements of coding, configuring, maintaining and repair work going on in the code.

***'Dev-ops' and continuous building: when making infrastructures becomes operating infrastructures***

These deeper fluxes are not easy to see in the nominalistic semiosis described above. They might be discerned however by selecting from amongst the millions of named repositories on Github some of those that relate not so much to the design and construction of software but to its deployment, maintenance and repair. Amidst the millions of repositories on Github, disrepair and abandonment are extremely common. Repositories that contain a few items uploaded at some date are extremely common, as are repositories that have not been touched for months or years. By contrast, some repositories on Github show almost continuous patterns of contribution (developers adding or modifying the code), and others, perhaps fewer again, are not only often modified but constantly copied.4 As we have seen earlier, the copying of a repository through the operation of forking or cloning is extremely common amongst developers, but the kind of copying in question here is not for the purposes of development by further coding. It is cloning for operational deployment. Such repositories serve different purposes than software development, or at least, they change how we might understand software development.

On the hand, they often contain specialised code focused on configuration of devices, systems, other software, services, and platforms. On the other hand, this specialized information is itself subject to generalization by virtue of its inclusion in a software repository. Configuration repositories are abound on Github because they allow the often intricate configuration details of contemporary software systems to be replicated quickly. Given that the scale of much information infrastructure is predicated on replicating the same thing multiple times, these configuration repositories reposition the management of infrastructure much closer to software development (e.g. a data-centre might be understood as a collection of identical computers linked together in ways that permit their processing to be coordinated and divided in many different ways; 'identical' here, however means managed through processes of replication.) Perhaps the most straightforward example of this somewhat recursive aspect of configuration would be the repository github/github. This repository name is not publicly visible on Github, yet its existence underpins the ongoing life and dynamism of Github as a social coding platform. As the several hundred developers employed by Github modify, add or repair elements of the platform itself, their changes flow into the operation of Github through continuous deployment routines that draw down changes from the github/Github. Devops -- development operations - blurs the line between making and running, between designing and maintaining. We could also identify much more mundane versions of devops-style processes occurring in many repositories on Github. The many repositories containing names such 'dot' or 'dotfiles' or 'conf' do something similar at the level of individual developers development configuration. That is, they ensure a common and stable development environment for software developers amidst the many different configuration options possible (in relation to editors, compilers, file systems and various elements of the infrastructure that have to be rendered locally operable so that development for larger scale deployments can occur).

'Dev-ops' or developing-operating continuously re-fashion infrastructures through continuous deployment. The increasing elision of any difference between making and operating is a striking feature of such infrastructures, and one that may affect the mode of existence of infrastructures more generally. Different infrastructural figures such as 'the platform,' 'the stack,' 'deployment' and 'build' intersect with various organisational, aesthetic, ethical and commercial regimes of engagement. A combination of different materials ranging from images, texts, cloud or distributed computing services, various database and search engines, through to code repositories with their groups of contributors comprise the loose but interwoven infrastructures typical of such platforms. They hold together in relational patterns that arise from recursive practice.

***Github itself - an island or an archipelago***

Where does this leave Github itself? If I have been arguing that the combinatorial delirium generated by NSA-O-Matic is infrastructurally relevant and resonant for us today because it shows how many things can be seen as compounded or composited almost typographically, does the specific infrastructure of a platform like Github belong to this process or simply support it? Is Github, then, a hub because it somehow has captured the nominalistic semiosis, or somehow made itself into the sinkhole of this semiosis? If Github attracts tremendous amounts of code (for instance, Google Corporation recently announced the closure of its own code repository service Google.code.com and migration of all code deposited there to Github (Google 2015); Microsoft Corporation has recently opened the source code of Windows, its main product, on Github, etc.), should we understand this as evidence of the success of Github in controlling the flows of code and coding work? To put the question more metaphorically, is Github a logistical success story (say like Singapore's role in container shipping) or a tremendous organic accumulation (say like Lagos in Nigeria, with its complicated flows of people, trade, oil and politics)? If we understand Github itself in terms of the nominalistic semiosis, then neither of these alternatives are quite right. Like many platforms, services, devices and systems, Github can be seen as an infrastructure, an infrastructure used in this case for software development and deployment. In this respect, Github is somewhat like a logistics hub. It socially containerises code and allows it to move much flexibly and speedily. Code and coding itself might be seen as re-globalised in their Github transformation. But this logistical management does not capture the tremendous morass of abandoned, rarely used, ephemeral, miscellaneous and incredibly specific code and coding found there.

***Conclusion***

I have been suggesting that in the case of software-affected infrastructures at least that names and naming practices help us practically engage with the full-stack complexities of contemporary infrastructures. Names in their linkages of people and devices, of 'dev' understood as a person, the software developer, and 'dev' as a common abbreviation for device, flow widely and deeply in contemporary software development. They attest to different and often overlapping movements. The imitative fluxes of names point to processes of replication and copying that propagate devices, protocols, standards and things in various configurations. I suggested that in contemporary information infrastructures, the form of the stack presents one organising axis of this propagation. But this layered architecture is viscously mixed in the concatenated names of many of the repositories on Github. Both the replication of particular forms - linux, jquery - and their mutating combination can be seen in the repository names.

Names not only provide a methodological traction on the infrastructural flux associated with software and coding practices, giving us some sense of the relative centrings and the predominant infrastructural rituals of our worlds. They operate generatively in several different senses. First of all, the very act of naming every collection of code, code that may or may not do something in the world, implies an increasing addressability of diverse bodies of code. This addressability supports and engenders what I have been calling, loosely following Peirce, nominalistic semiosis. Second, the expansion of what is called a code repository to include almost any aspect of the structure, configuration, documentation and coordination of work associated with contemporary infrastructures means that the same regime of addressing in order to operate can generalization. Hence infrastructures increasingly unfurl by virtue of an ostensibly superficial nominalistic pragmatism. Third, this nominalistic semiosis is somewhat indifferent to distinctions between making and using, between design and deployment, and indeed to differences between what people and devices do. Across all of these distinctions, naming and calling by name operate interchangeably. This indifference or re-drawn difference between people and infrastructures generates a constant unfurling deployment in time and space. It lends itself to replication and variation, to changes in scale and variations in configuration. Devops epitomises one highly distilled version of the nominalistic semiosis of infrastructure.

Following on from both STS work on infrastructures and media theory accounts of embodied time-space, it makes sense to see these unfurlings as also generating spaces and times, and not simply located in space and time. Unfurling is result of the relatively heterogeneous and diverse scales stacked on each other to generate relative durations and extended replications. Yet this unfurling makes times and spaces as well since it derives from new forms of holding-together assayed by naming and addressing named things. This unfurling may be very predictable, and it may also be shaped by imaginaries (of the cloud, of mobile communication, of pervasive computing), but it will also bring to light new forms of holding together. The continuous building of devops would be one example of this.

***References***

Bowker, G. C. 2005. *Memory Practices in the Sciences*. MIT Press Cambridge, MA.

Callon, Michel, Jean-Pierre Courtial, William A. Turner, and Serge Bauin. 1983. “From Translations to Problematic Networks: An Introduction to Co-Word Analysis.” *Social Science Information* 22 (2): 191–235. doi:[10.1177/053901883022002003](http://dx.doi.org/10.1177/053901883022002003).

CJFE. 2015. “Snowden Digital Surveillance Archive.” <https://snowdenarchive.cjfe.org/greenstone/cgi-bin/library.cgi>.

Coleman, Gabriella. 2012. *Coding Freedom: The Ethics and Aesthetics of Hacking*. Princeton N.J.: Princeton University Press.

Edwards, P. N., M. S. Mayernik, A. L. Batcheller, G. C. Bowker, and C. L. Borgman. 2011. “Science Friction: Data, Metadata, and Collaboration.” *Social Studies of Science* 41 (5): 667–90. <http://sss.sagepub.com/content/41/5/667.short>.

Foucault, Michel. 1972. “The Archaeology of Knowledge and the Discourse on Language (Trans: A. Sheridan).” *Pantheon, New York*.

Git. 2014. “Git.” <http://git-scm.com/>.

Google. 2015. “Bidding Farewell to Google Code. Google Open Source Blog.” <http://google-opensource.blogspot.com/2015/03/farewell-to-google-code.html>.

Graham, Stephen, and Simon Marvin. 2001. *Splintering Urbanism : Networked Infrastructures, Technological Mobilities and the Urban Condition*. London: Routledge.

John, Nicholas A. 2013. “Sharing and Web 2.0: The Emergence of a Keyword.” *New Media & Society* 15 (2): 167–82. doi:[10.1177/1461444812450684](http://dx.doi.org/10.1177/1461444812450684).

Kelty, Christopher. 2008. *Two Bits: The Cultural Significance of Free Software*. Duke University Press.

Mackenzie, Adrian. 2006. *Cutting Code: Software and Sociality*. Digital Formations. New York: Peter Lang.

Netcraft. 2015. “January 2015 Web Server Survey | Netcraft.” <http://news.netcraft.com/archives/2015/01/15/january-2015-web-server-survey.html>.

Suchman, Lucy. 2006. *Human and Machine Reconfigurations: Plans and Situated Actions*. 2nd ed. Cambridge University Press.

Takhteyev, Yuri, and Andrew Hilts. 2010. *Investigating the Geography of Open Source Software Through GitHub*. <http://takhteyev.org/papers/Takhteyev-Hilts-2010.pdf>.

Thrift, Nigel. 2014. “The ‘Sentient’ City and What It May Portend.” *Big Data & Society* 1 (1): 2053951714532241. doi:[10.1177/2053951714532241](http://dx.doi.org/10.1177/2053951714532241).

Thung, Ferdian, Tegawendé F. Bissyandé, David Lo, and Lingxiao Jiang. 2013. “Network Structure of Social Coding in GitHub.” In *Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on*, 323–26. IEEE. <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6498480>.