The Social (Re)Production of Architecture

Author(s) Petrescu, Doina; Trogal, Kim

Imprint Taylor and Francis, 2017

ISBN 9781138859487, 9781138859494, 9781315717180, 9781317509233, 9781317509219

Permalink https://books.scholarsportal.info/uri/ebooks/ebooks4/taylorandfrancis4/2018-06-08/7/9781317509233

9 — SOFTWARE AND SPATIAL PRACTICE: THE SOCIAL (CO)PRODUCTION OF SOFTWARE OR SOFTWARE FOR SOCIAL (CO)PRODUCTION?

Phil Langley

The practice of architecture is conditioned by our technology, from drawing boards and pencils through laptops and screens, to sensors and networks, yet the mediating nature of this technology often goes unacknowledged. Perhaps this relationship with the 'tools of our imagination' (Piedmont-Palladino, 2007) was easier to ignore in the past, but the need to engage with the effects of software on our behaviours (and vice versa) should no longer be in doubt. While the social nature of the scientific laboratory has been described by sociologists of science such as Latour, and the significance of non-human devices in the production of knowledge identified and documented (Latour and Woolgar, 1986), the same is not true of spatial design practice and its relationship with digital technology. We have definitively moved beyond the false paradigm of 'Computer Aided Design', in which skeumorphic[1] software packages echoed the functionality and reinforced the behaviour of the drawing board, but where we have moved to is less certain.

The current discourse around spatial design and digital technology includes speculations on smart cities (and, by dubious extension, smart citizens) and the rise of the so-called 'internet of things'. Each of these two strands focuses on a 'near future' vision of ubiquitous computing in our urban environments that, in their most optimistic form, suggest significant reductions in waste and energy use, increased efficiency of transportation and communication infrastructures and even of democracy itself. Such optimism, should of course, be treated with caution – not least as it poses the question, by whose measure can these changes be described as 'improvements'? But

1 The term skeumorphic relates to objects whose design has been principally derived from the required functionality of an earlier object. A more general example of skeumorphism would be a digital watch that displays on its screen the rotating hands of an analogue watch.

software is already central to the design and documentation of buildings and cities. It is used for 3D, 4D (even 5D)[2] modelling as well as for complex simulations and analyses of virtual environments. At the same time, it is becoming more important in the operation of those environments once they have become material. Most significantly here, the use of particular software, at certain stages of design and operation is becoming more and more prescriptive, placing significant demands on building designers and users.[3] The practice of spatial production is firmly situated in a new digital reality in which software can no longer be seen as merely a tool.

Software itself is always social, either actively so through the interactions of developers, or more passively through the historic layering of code which makes use of previously written code.[4] I am concerned specifically with using an idea of 'social coding' as a way of challenging mainstream ideas of what software in spatial design practice should be and what software should do. We need to develop alternative approaches and explore new relationships between ourselves and our technologies. I want to examine software itself as a legitimate site of study – what is software actually made 'of'? – and in particular the culture of software (Fuller, 2003, 2008). What is the social agency of software and how can this be used in both the social (co)production of software and the development of software for social (co)production? In order to answer this, I will use the ontological framework of the 'post-human', and in particular Donna Haraway's approach (Haraway, 2003), as a way of re-casting ourselves in relation to our technologies. Additionally, I will look to the radical development of 'queer technology' (Blas, 2006, 2008), as a design methodology

2 3D is, of course, model geometry and 4D is time simulations carried out using that geometry. '5D' represents another layer of socio-economic data applied to the model and typically refers to cost modelling.

3 I am writing this in the context of a significant change to mainstream architectural practice in the UK, which is in the process of being implemented, the effect of which will be to determine the means of digital production within architectural offices and, by extension, schools of architecture throughout the country. In April 2016, the UK government implemented procurement standards that require all projects funded by central government to be delivered using complex BIM (Building Information Modelling) software that is currently only available as a closed, proprietary product. This will determine the way in which much of the built environment will be designed, described and documented. Leaving aside the detailed implications of prescribing the use of such software specifically, which is worthy of a separate discussion, this enshrining of behaviour highlights precisely the fallacy that software is merely a tool. This is software as a method of control. See www.bimtaskgroup.org

4 For an example of 'Social coding', see the widely popular coding sharing platform 'github', available at: https://github.com

for software and technology, which encourages a 'destabilisation' of the normative binaries and dichotomies. These two strands set an alternative horizon for our engagements with technology and through an example of my own practice, this chapter aims to explore these implications.

DIGITAL COMPANIONS

In Haraway's Cyborg Manifesto, the ontological notions of 'human' and 'non-human' are destabilised and the oppositional distinctions between human (or animal) and machine instead become 'leaky' (Haraway, 1991). These cyborg formations imply blurriness at the boundaries of categorisation and suggest a complex, co-evolutionary process rather than a simple combinatory one. Haraway's cyborg is not created though mechanical couplings between entities from closed categories, but rather the dynamic topological relationships across blurred edges that create new, but inherently unstable (virtually, materially, temporarily) entities. Haraway develops her concept of the cyborg into an idea of a 'companion species' that represents a further 'synthetic de-centralising' of the natural and the artificial (Haraway, 2003). The cyborg destabilises any notion of categorisation to such an extent that 'species' can no longer be treated in isolation and requires a particular understanding of co-evolution:

> *It is a mistake to see the alterations of dog's bodies and minds as biological and the changes in human bodies and lives, for example, in the emergence of herding or agricultural societies as cultural, and so not about co-evolution.*
> (Haraway, 2003: 31)

For Haraway, the fate of the 'dog' and the 'human' have become so intertwined that any hierarchy between the two is flawed and the individuation of species becomes redundant. Furthermore, her notion of 'co-evolution' requires a non-deterministic approach to such relationships. Haraway's cyborgs and companion species outline a post-human condition in which we are placed in a dynamic topological relationship with/as 'companion species' that we make and remake, at the same time that they make and remake us. While Haraway is specific about dogs in her manifesto, it seems to me to be interesting, relevant and useful to extend this concept to an idea of a 'digital companion species'. Such a digital companion would not be an embedded 'productised' device or 'assistant', but instead would

be an ever-changing presence, over which we can exert influence and which, in turn, can influence us. A 'digital companion species', one that follows a process of co-evolution, would be more reflective of the variable agencies of software.

THE SOCIAL AGENCY OF SOFTWARE

> *Our machines are disturbingly lively and we ourselves frighteningly inert.*
> (Haraway, 1991: 152)

Everything I do appears to be somehow mediated by digital technology and not just the kind of cognitive expansion afforded by personal networked devices such as smart phones. In my professional work, the software I use or create does not merely act as a tool offering me a way to carry out an already defined task in a more efficient way, it does not provide a short cut to a pre-known destination. It also carries its own logic that mediates my activity and in doing so exhibits its own agency. At it most simplistic, the software can be seen as having 'secondary agency', extending that of its creator into the realm of the user, so that the creator is able to act as a 'guiding hand' to those that deploy the software. This description is, at first glance, an attractive one. It allows me, first, as the user of software, to critique the expectations of its use in practice and, second, as a creator of software, implies that I can 'influence' the practice of others. But it isn't that simple.

The notion of the 'secondary agency' of software implies a deterministic relationship between creator and user – or, as these terms are modified in Mackenzie's ontology of software, originator and recipient (Mackenzie, 2006).[5] Instead, the agency of software is relational – it has a variable capacity to mesh with other actors within new contexts (Mackenzie, 2006; Kitchin and Dodge, 2011). Kitchin and Dodge use climate change modelling to illustrate this definition of the agency of software. The complex simulations carried out to predict the effect on the Earth's atmosphere – a huge, collaborative scientific undertaking – are distilled into single digit temperature increases across the whole globe, that are deemed to be either acceptable or not during political negotiations. In this case, the simulation model – the software – is not carrying the agency of its creators. Instead it is demonstrating its capacity to affect as part of a

5 The full ontology is as follows: 1. Code as index; 2. Originator; 3. Recipient; 4. Prototype.

relational system.[6] 'The models analyse the world, the world responds to the models' (Kitchin and Dodge, 2011: 30).

Mackenzie describes the edges of each part of his software ontology as fuzzy and states that what is important is that 'the patterns of relations that unfold in the neighbourhood of software are agential' (Mackenzie, 2006: 17). That is to say, that each of the entities in the ontology may act on any other at any given moment, depending on circumstance. Furthermore, this describes software not as a stable entity with its own discrete agency, but rather as something that only ever exists as part of a larger formation that includes both its past 'origins' and its future 'destinations'. Furthermore, software doesn't just appear, as Mackenzie states:

> *Someone or something codes it; there is an originator. Whether the originator is a programmer, webmaster, corporation, software engineer, team, hacker or scripter, and regardless of whether the originator's existence can be forgotten, sanctified or criminalized, software originates somewhere.*
> (ibid.: 14)

In this way, the software is not acting on something or someone, instead its effects are co-produced between the fluid, interchanging roles of 'originator' and 'recipient'. I would go further in the definition of 'originator' to include those whose use of proprietary software becomes a tacit approval of its functionality and in some way perpetuates it. By this description, we are all originators of software and we are all already involved in its (co)production.

In her book, *Close to the Machine*, Ullman provides a first-person account of the messy nature of software development and, by extension, software. Her narrative, based on her own experiences as a professional programmer, reveals the relational agency of software through the social intra-actions of the 'code', 'originators', 'receivers' and 'prototypes', and she provides specific testimony on the social nature of software. Ullman talks particularly about writing code, when it is at its most unstable, and in doing so offers an alternative description of the 'discipline' of programming as a dynamic (rather than procedural) process, that has no meaningful beginning or end. 'It has

6 This illustration of the agency of software can also be seen at the scale of a building, where complex environmental simulations carried out at the design stage to predict the internal conditions are simplified to single letter performance ratings – A, B, C, etc. – determined by regulatory bodies, denoting 'overall' success or failure.

occurred to me that if people really knew how software got written I'm not sure they'd give their money to a bank, or get on an airplane again'[7] (Ullman, 2013: 2).

Code is always on the 'verge of disappearance' or collapse, precisely because it is relational and co-evolving. It barely works and without nurture and care, it will begin to deteriorate and break down. Functionality is lost as the 'ecosystem' which it inhabits changes and it becomes less and less intelligible as the code loses contact with those who created and understood its unique idiosyncrasies.

'QUEER' TECHNOLOGY

The F/LOSS (Free/Libre Open Source Software) movement, in which the term 'free' refers not to the cost but the users' freedom to study, modify and distribute the software, provides a broad framework for other ways of producing software (and has, of course, influenced many other discussions on copyright and ownership). This breadth includes both the creation of free alternatives to closed, propriety software[8] as well as more radical propositions. Artist and writer Zach Blas has made one such radical proposition which he describes as Queer Technologies (Blas, 2006, 2008). Blas's approach has developed from the wider discourse of queer theory, in which the term 'queering' is used not only in relation to groups such as gay, lesbian and transgender but can also be understood as a performative act against dominant perceptions and normative systems. For Blas, this includes a design methodology that embraces 'uselessness' as a way of challenging the ways in which software is considered to 'work' and is able to disrupt the normative binary by working across them. This queering of software is achieved through questioning the very function of functionality. As Blas says:

> *I think Queer Technologies wants to work in the interstices of useful and useless, or to find new uses through the useless. Importantly, this is not about deconstruction, it is about use, about doing something, experimenting with new ways of doing and making things happen.*
> (Interview with Zach Blas, n.d.)

7 I would also add 'design a building' to Ullman's statement.

8 Examples include the image editing software GIMP, which offers an alternative to Adobe Photoshop (www.gimp.org/index.html) or Open Office in place of Microsoft Office (www.openoffice.org).

Artist Željko Blaće works with this approach and takes the queer operating system as a paradigm for proposing other ways of doing software that challenge the normative systems of productivity and digital technology. Blaće's project for a Queer OS is an ongoing exploration of speculative proposals and prototypes, carried out through a series of collaborative events and workshops. Blaće places significance on the act of creative inquiry (rather than the technical activity of writing code, for example) and thus provides a platform for wider participation in the social, economic and political debates that surround software.[9]

The queering of software offers a strong design approach for those working with, as well as on, software. Rather than seeking to chase the supposed functionality of proprietary norms (something that could be said of F/LOSS projects such as Open Office), queerness allows for a more creative process of exploration that addresses the social nature of software and goes further than the very general aims of the F/LOSS movement and provides a specific software design methodology.

'SOCIAL' MEDIA

While these approaches might help inform 'small-scale' technological interventions such as coding on a personal computer, for instance, how can they help us also engage with the kinds of networked computing that have become so pervasive? More so than so-called 'cloud' storage systems for music or video files which are typically operated 'on demand' by the user, social media systems represent a perverted kind of digital companion that is always on, always operating, always connecting. Facebook, Twitter and other social media software provide not only a platform for our own, directed communication, but a means of aggregating content from others, sometimes our known contacts but also from unsolicited sources, such as advertisers or other 'curated' content.[10] Its function as a direct communication

9 A 2014 workshop by Željko Blaće on the Queer OS was held in Brussels in 2014, hosted by arts-lab *Constant*, as part of their GenderBlending workshop http://constantvzw.org/site/-GenderBlending,190-.html. The discussion during the workshop included general proposals to change the skeumorphic features of normative operating systems such as files and folders as well as specific queer 'functionality' such as discontinuous communication systems (i.e. receiving a message via one account and replying via another), as a way of combating digital surveillance.

10 Social media corporations are also not averse to 'curating' this content in order to manipulate users. Facebook was revealed to have managed the flow of positive and negative news stories to users' news feeds in order to control their emotions: www.theguardian.com/technology/2014/jun/29/facebook-users-emotions-news-feeds

platform has made social media an often reported feature of political uprising and protest in many countries around the world (Castells, 2012; Gerbaudo, 2012). Manuel Castells claims that networks, which for him are the dominant mode of organisation of our society, are controlled by those who can program the networks and can switch the networks. Here, to program is to set the goals of the network, whereas to switch is to connect different networks and share those goals. In his words:

> *If power is executed by programming and switching networks, then counter-power, the deliberate attempt to change power relationships, is enabled by reprogramming networks around alternative interests and values and/or disrupting the dominant switches while switching networks of resistance and social change.*
> (Castells, 2012: 9)

Castells advocates a direct challenge to the mechanisms of power through the re-programming of its networks and disrupting dominant switches, urging us to occupy the medium of communication. Castells even goes as far as to claim the Occupy movement was being born digital, which suggests that characteristics of these protest movements have been irrecoverably altered by the use of social media.

These broad claims of the significance of social media in the facilitation of collective action are, perhaps, overly simplistic (Gerbaudo, 2012). Amidst the understandable optimism that many protest movements have created (as well as the undoubted additional 'functionality' that global communication networks in general provide), there is, at least for me, an accompanying concern at the suggestion that the success of such movements may rest on the continued use of a technology that lies so far outside of the influence or control – commercially, politically or materially – of the protagonists.[11] And, of course, the widespread use of social media by those who wish to challenge existing power structures has led to increased scrutiny of digital communication by governments worldwide. The ability of global

11 One striking example that makes visible this power imbalance is a tweet sent during the 'Green Revolution' in Iran, in 2009. 'ALL internet and mobile networks are cut. We ask everyone in Tehran to go onto their rooftops and shout ALAHO AKBAR in protest #IranElection' https://twitter.com/mousavi1388/status/2156978753. The message was sent by a supporter of the reformist politician Houssein Mousavi who stood for election during the contested 2009 elections that precipitated the uprising. The 140-character message encapsulates the fragility of our access and the impermanence of these networks during what was, ultimately, an unsuccessful opposition movement.

security services to spy on the mass of emails, calls, messages, tweets, likes, favourites, etc. has brought into sharp focus questions around our relationship with such platforms, the commercial interests that supply them, and the government agencies that oversee them. Nevertheless, there is something appealing in Castells's optimism of the potential for social media, and the direct link he makes to the occupation of the material and the virtual: 'They build their projects by sharing their experiences. They subvert the practice of communication as usual by occupying the medium and creating the message' (Castells, 2012).

@SIMULATIONBOT

Using the conceptual framework of post-human de-centring of the natural and synthetic, alongside the social agency of code and using some of the aspects of queering technology, I would like to offer a prototype for software for social (co)production, in terms of spatial design practice. The @simulationBot project is an attempt to advance an alternative strand of open source software for spatial design that does not attempt to duplicate the functionality of existing, proprietary platforms such as those used for complex geometry modelling or data management. Instead, I am suggesting other types of interface and interaction with design and software that develop, rather literally, the idea of a (digital) companion species and in some way attempt to 'occupy the medium'.

The @simulationBot is a kind of 'twitter bot',[12] a computer program that automatically 'tweets' in response to certain stimuli. It is a prototype project that appropriates three familiar characteristics of the micro-blogging platform Twitter – 'liveness', 'hashtagging' and 'geo-location' – to propose an alternative idea of software for spatial design (Figures 9.1 and 9.2).

12 See http://en.wikipedia.org/wiki/Twitterbot

Figure 9.1 Photos by workshop participants

Figure 9.2 Design 'hacks' by workshop participants

A twitter bot, which typically runs continuously on web servers, can be used for various purposes, including spamming users. Depending on their complexity, the twitter bot can respond in many different ways but it typically is not based on any AI (artificial intelligence) system and its behaviour is mostly 'hard-coded',[13] for example, bots may automatically re-tweet the post of another user. More complex bots tweet about events on other platforms, for example, @parliamentedits posts any changes to Wikipedia pages made from an IP address inside the UK Parliament building.[14] Regardless of the specifics of the behaviour, the bots offer a novel way of 'occupying the medium', working within the constraints for the platform as con-

13 Hard coding refers to the act of embedding data and/or data structures into a program, rather than being able to generate it dynamically. For example, the file path to a user's documents folder on a personal computer would be hard-coded into the operating system. While it is not necessarily 'bad' (and is, in fact often necessary), hard coding can result in fixed software behaviours.

14 See https://twitter.com/parliamentedits, or https://gist.github.com/Jonty/aabb42ab31d970dfb447, or www.theguardian.com/technology/2014/jul/30/how-to-find-out-when-uk-politician-edits-wikipedia-page

trolled by their corporate owners, but also extending the functionality of those platforms beyond that which its originators had intended. In doing so, twitter bots are able to appropriate not only the software but also the networked hardware that support them – data centres, mobile and wireless communications and the personal computing devices of Twitter users. The bots are still fragile – they can only function while Twitter allows programmers access to their platform through the API[15] and as long as the platform itself is left switched on – but it is still a more extensive 'reprogramming' of the network in which not only the message is altered, but also the medium.

The behaviour of my own digital companion – @simulationBot[16] – was developed to employ not only the underlying functionality of the Twitter platform, but was also designed in opposition to other characteristics that have developed as part of the wider exploitation of Twitter. The technology of social media platforms provides a very simple method of 'content aggregation', but they do not, in themselves, provide a reliable format for the content itself because we as users, don't view ourselves as 'content' generators.

It is common to see, both in academic research as well as in contemporary news media, visualisations of Twitter data, sometimes represented on a map, or aerial photograph. This is typically produced from a static data set, 'scraped' from the posts of unsuspecting users some time after an event has occurred. In this kind of arrangement, the Twitter users are merely passive suppliers of data to an unknown outside observer, rather than active participants in the process. As a consequence of this imbalance, the data set itself is severely undermined. While there is no such thing as a 'complete data set', this kind of data set is particularly flawed when represented in a generalised way. At a population scale, it is unlikely that the demographics of all Twitter account holders will ever be 'representative' and nor will those tweeting at any given moment be representative either. And at the scale of a single tweet, it requires active use of the geo-locative functionality of each user's device and rigorous use of the hashtag mechanism to make a tweet in any way machine-readable.

---

15 The API, or the Application Programming Interface, is a set of protocols that determine how publicly exposed software components can interact and how developers can use them to create their own outcomes. The API is controlled by the principal 'originators' of a piece of software and the depth of its functionality can vary widely. Furthermore, changes to the API can result in previously functioning software becoming obsolete.

16 The twitter bot simBotBETA @simulationBot was created using www.processing.org and the libraries www.twitter4j.org and http://unfoldingmaps.org/. Source code for @simulationBot can be found at www.github.com/phiLangley

The @simulationBot project is about altering our interaction with the software of Twitter in order to create an alternative 'social' assembly and was created using these fundamental principles:

— The 'digital companion' is not merely an outsider and should be an active member of the social network.
— The 'digital companion' must be, in some way, present and visible, rather than remote.
— The data has to be made actively and not collected passively.
— The data must be re-presented in real time.

The @simulationBot was developed for, and during, a series of workshops intended to question the nature of software and code in spatial design practice.[17] In the final experiment participants were asked to explore the city centre of Sheffield and propose 'hacks' – which are a sort of 'micro' design intervention – which they would tweet about, using photos and text, as well as hashtags and their GPS location (Figure 9.2). The tweets were collated in real time by the @simulationBot, which produced a live map of data, indicating the individual and group activity that was projected at the event venue as well as re-tweeted to the dispersed participants. So, the @simulationBot acts as our digital agent within the system. As members of the group send tweets from the locations across the city, they receive notification from @simulationBot about the activities of other members of the group. The map of the tweets was not only built in real time, but also shared in real time, as snapshots were shared in the communications of the @simulationBot (Figure 9.3).

While operating in this way the @simulationBot is no longer only exploiting the infrastructure of social media – GPS-enabled devices, sending and storing text and media across communication networks – but rather is beginning to modify it. The usual method of directed interface from the service provider – targeted ads, curated content, and so on – is replaced with a 'digital companion' participating in the activity as well as enabling it.

17 The simBot workshop was part of 'The Whole School Event – Designs on our City' which took place in February 2014, and was organised by the Sheffield School of Architecture (SSoA), University of Sheffield. The workshop, entitled 'open data in the city' was carried out with Dr Mark Meagher, of SSoA, and included student participants from across SSoA. See https://architecture.dept.shef.ac.uk/ssoa_news/?p=1812

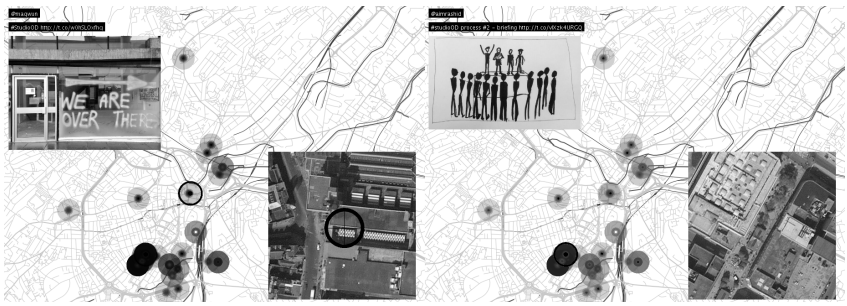

Figure 9.3 Maps created by @simulationBot and Phil Langley

SOFTWARE AND SPATIAL PRACTICE

It is clear to me that, as architects, our relationship with our technology is central to our practice. The unstable agencies of software show that we cannot consider it a neutral tool in the production of space. The behaviour of software affects the process of design and, similarly, software should be affected by the type of design process that we want to adopt. Software cannot simply be deployed in order to realise a designer's wishes. Instead, to work with software and technology should be a reflexive practice in which the relations between designer and the software are made and remade. The idea of social (co)production of space is not yet part of mainstream architectural practice and it seems unlikely to me that software that is developed for that mainstream would be wholly applicable to this other approach. The @simulationBot is a very simple attempt to propose other types of software that may be (co)produced for spatial design. We need alternatives that are not just different in terms of access – 'open' rather than 'closed' – but are distinctive in functionality. The very notion of functionality must be re-examined and the queering of technology offers a convincing design methodology for both understanding and critiquing existing software and proposing others. Code is an unstable material, from which sturdy algorithms and software emerge. While stability is seen as a prerequisite for software to be successful, perhaps we shouldn't aspire to it. The problematic hardware and software assemblies of networked communication technology, for instance, represent a level of stability that we cannot realistically hope to replicate, so perhaps we shouldn't even try. Rather, it is in the messiness of code where we can co-evolve with our own digital companion species, species which have agency, which have behaviour, which 'affect' and are 'affected'.

REFERENCES

Blas, Z. (2006) *What is Queer Technology?* Available at: www.zachblas.info/publications_materials/whatisqueertechnology_zachblas_2006.pdf

(2008) *Gay Bombs: User's Manual, Queer Technologies*. Available at: http://www.zachblas.info/wp-content/uploads/2016/03/GB_users-manual_web-version.pdf

(n.d.) Interview with Zach Blas. Available at: http://rhizome.org/editorial/2010/aug/18/interview-with-zach-blas/ (accessed 28 February 2015).

Castells, M. (2012) *Networks of Outrage and Hope: Social Movements in the Internet Age*, Cambridge: Polity Press.

Fuller, M. (2003) *Behind the Blip: Essays on the Culture of Software*, Brooklyn, NY: Autonomedia.

(2008) *Software Studies: A Lexicon*, Cambridge, MA: MIT Press.

Gerbaudo, P. (2012) *Tweets and the Streets: Social Media and Contemporary Activism*, London: Pluto Press.

Haraway, D. (1991) *Simians, Cyborgs and Women: The Reinvention of Nature*, London: Free Association Books.

(2003) *The Companion Species Manifesto: Dogs, People and Significant Otherness*, 2nd edn, Chicago: University of Chicago Press.

Kitchin, R. and Dodge, M. (2011) *Code/Space Software and Everyday Life*, Cambridge, MA: MIT Press. Available at: http://site.ebrary.com/id/10479192

Latour, B. and Woolgar, S. (1986) *Laboratory Life: The Construction of Scientific Facts*, ed. J. Salk, Princeton, NJ: Princeton University Press.

Mackenzie, A. (2006) *Cutting Code: Software and Sociality*, New York: Peter Lang.

Piedmont-Palladino, S. (2007) *Tools of the Imagination: Drawing Tools and Technologies from the Eighteenth Century to the Present*, New York: Princeton Architectural Press.

Ullman, E. (2013) *Close to the Machine: Technophilia and its Discontents*, London: Pushkin Press.