Living Books About Life OPEN HUMANITIES PRESS JISC

Life in Code and Software:

Mediated Life in a Complex Computational Ecology

edited by David Berry

Introduction

This book explores the relationship between living, code and software. Technologies of code and software increasingly make up an important part of our urban environment. Indeed, their reach stretches to even quite remote areas of the world. *Life in Code and Software* introduces and explores the way in which code and software are becoming the conditions of possibility for human living, crucially forming a computational ecology, made up of disparate software ecologies, that we inhabit. As such we need to take account of this new computational environment and think about how today we live in a highly mediated, code-based world. That is, we live in a world where computational concepts and ideas are foundational, or ontological. Here, code and software become the paradigmatic forms of knowing and doing - to the extent that other candidates for this role, such as air, the economy, evolution, the environment, satellites, and so forth, are understood and explained through computational concepts and

categories.

Certainly, computer code and software are not merely mechanisms; they represent an extremely rich form of media. They differ from previous instantiations of media in that they are highly processual. They can also have agency delegated to them, which they can then prescribe back onto other actors, but which also remains within the purview of humans to seek to understand. As Kitchin argues:

> across a diverse set of everyday tasks, domestic chores, work, shopping, travelling, communicating, governing, and policing, software makes a difference to how social, spatial, and economic life takes place. Such is software's capacities and growing pervasiveness that some analysts predict that we are entering a new phase of 'everyware' (Greenfield, 2006); that is, computational power will be distributed and available at any point on the planet. (Kitchin, 2011: 945)

The deeply interactive characteristic of code and software makes computational media highly plastic for use in everyday life, and as such it has been highly successful in penetrating more and more into the lifeworld. Digital code/software has created, and continues to create, specific tensions in relation to old media forms, such as the disruption it has produced in the print, music and film industries, as well as problems for managing and spectacularising the relations of the public to the entertainment industry and politics. This is something that relates to the

interests of the previous century's critical theorists, particularly their concern with the liquidation of individuality and the homogenization of culture. Nonetheless, there is also held to be a radical, if not revolutionary kernel within the softwarization project. It is a potential that is understood as relating to the relative affordance code/software appears to provide for autonomous individuals within networks of association to share information and communicate, often theorised as a form of network politics. Indeed, as Deuze *et al* have argued:

> Considering the current opportunity a media life gives people to create multiple versions of themselves and others, and to endlessly redact themselves (as someone does with his/her profile on an online dating site in order to produce better matches), we now have entered a time where… we can in fact see ourselves live, become cognizant about how our lifeworld is 'a world of artifice, of bending, adapting, of fiction, vanity, a world that has meaning and value only for the man who is its deviser' [Pirandello 1990, 39]. But this is not an atomized, fragmented, and depressing world, or it does not have to be such a world. (Deuze, Blank, and Speers, 2012)

I want to understand the ecology in computational ecology here as a broad concept related to the environmental habitus of both human and non-human actors. My aim in doing so is to explore changes that are made possible by the installation of code/software via computational devices, streams,

clouds, or networks. This is what Mitcham calls a 'new ecology of artifice' (1998: 43). The proliferation of contrivances that are computationally based is truly breathtaking - each year we are provided with fresh statistics that demonstrate just how profound the new computational world is. For example, 427 million Europeans (or 65 percent) use the Internet and more than 9 in 10 European Internet users reading news online (Wauters, 2012). These computationally based devices, of course, are not static, nor are they mute, and their interconnections, communications, operation, effects and usage remain to be properly studied. It is a task that is made all the more difficult: both by the staggering rate of change, thanks to the underlying hardware technologies, which are becoming ever smaller, more compact, more powerful and less power-hungry; and by the increasing complexity, power, range and intelligence of the software that powers it.

These computational devices, particularly mobile forms, also enable the assemblage of the new social ontologies and the corresponding social epistemologies that we have increasingly come to take for granted in computational society, including Wikipedia, Facebook, and Twitter. The extent to which computational devices, and the computational principles on which they are based and from which they draw their power, have permeated the way we use and develop knowledges in everyday life continues to expand driven by the network effects of digital media. The ability to call up information instantly from a mobile device, combine it with

others, subject it to debate and critique through real-time social networks, and then edit, post and distribute it worldwide would be incredible if it hadn't become so mundane (see, for example, Hall, 2011).

Today it should hardly come as a surprise that code/software lies as a mediator between ourselves and our corporeal experiences. Code/software are the materialisation of computationality, in that they are the medium through which structural features of computation are realised and mediated. For example, code/software disconnects the physical world from a direct coupling with our physicality, whilst managing a looser softwarized transmission system (see also Parikka, 2012). Called 'fly-by-wire' in aircraft design, in reality fly-by-wire is the condition of the computational environment we increasingly experience, and I elsewhere term *computationality* (Berry, 2011). This is a highly mediated existence and has been a growing feature of the (post) modern world. Whilst many objects remain firmly material and within our grasp, it is easy to see how a more softwarized simulacra lies just beyond the horizon. Not that software isn't material, of course. Certainly, it is embedded in physical objects and the physical environment and requires a material carrier to function at all. Nonetheless, the materiality of software is without a doubt *differently* material, more *tenuously* material, almost less *materially material*. That is, the material form of code/software is difficult to theorise and understand due to its perceived invisibility or ethereality, while having

concrete effects nonetheless. This is partly, it has to be said, due to software's increasing tendency to hide its depths behind glass rectangular squares which yield only to certain prescribed forms of touch-based interfaces. Here I am thinking both of physical keyboards and trackpads, as much as haptic touch interfaces, like those found in the iPad and other tablet computers. Another way of putting this, as N. Katherine Hayles (2004) has accurately observed, is that print is flat and code is deep - although it is useful to note that some theorists, such as Frabetti (2010), have problematised Hayles' understanding of code, print, and materiality.

Web Bugs, Beacons, and Trackers

Some examples will help to demonstrate how this code-based world is increasingly being spun around us. Firstly, we might consider the growing phenomena of what are called 'web bugs' (also known as 'web beacons'); that is, computer programming code that is embedded in seemingly benign surfaces, but which is actively and covertly collecting data and information about us.[1] As Madrigal explains:

> This morning, if you opened your browser and went to NYTimes.com, an amazing thing happened in the milliseconds between your click and when the news about North Korea and James Murdoch appeared on your screen. Data from this single visit was sent to 10 different companies, including Microsoft and Google

> subsidiaries, a gaggle of traffic-logging sites, and other, smaller ad firms. Nearly instantaneously, these companies can log your visit, place ads tailored for your eyes specifically, and add to the ever-growing online file about you… the list of companies that tracked my movements on the Internet in one recent 36-hour period of standard web surfing: Acerno. Adara Media. Adblade. Adbrite. ADC Onion. Adchemy. ADiFY. AdMeld. Adtech. Aggregate Knowledge. AlmondNet. Aperture. AppNexus. Atlas. Audience Science… And that's just the As. My complete list includes 105 companies, and there are dozens more than that in existence. (Madrigal, 2012).

Web bugs are automated data collection agents that are secretly included in the web pages that we browse. Often held within a tiny one-pixel frame or image, which is therefore far too small for the naked eye to see, they execute code to secrete cookies onto your computer so that they can track user behavior, and send various information about the user back to their servers.

Originally designed as 'HTTP state management mechanisms' in the early 1990s, these data storage processes were designed to enable webpages and sites to store the current collection of data about a user, or what is called 'State' in computer science. Known as 'web bugs for web 1.0' (Dobias, 2010: 245), they were aimed at allowing website designers to implement some element of memory about a user,

such as a current shopping basket, preferences, or username. It was a small step for companies to see the potential of monitoring user behaviour by leaving tracking information about browsing, purchasing and clicking behaviour through the use of these early 'cookies'.[2] The ability of algorithms to track behaviour, and collect data and information about users raises important privacy implications, but it also facilitates the rise of so-called behaviour marketing and nudges (for a behaviourist approach see Eyal, 2012). These technologies have become much more sophisticated in the light of Web 2.0 technologies and developments in hardware and software: in effect, web bugs for web 2.0 (Dobias, 2010: 245).

Fortunately, we are seeing the creation of a number of useful software projects to allow us to track the trackers: Collusion, Foxtracks and Ghostery, for example.[3] If we look at the Ghostery log for the ChartBeat company it is described as:

> Provid[ing] real-time analytics to web sites and blogs. The interface tracks visitors, load times, and referring sites on a minute-by-minute basis. This allows real-time engagement with users giving publishers an opportunity to respond to social media events as they happen. ChartBeat also supports mobile technology through APIs. (Ghostery, 2012b)

Web bugs perform these analytics by running code run in the browser without the knowledge of the user, which if it should be observed, looks extremely complicated.[4] Here are two early web bugs (web 1.0) collected by the Electronic Frontier Foundation (EFF) (1999):

- <img src="(http://ad.doubleclick.net/ad/pixel.quicken/NEW)" width=1 height=1 border=0>
- <IMG WIDTH=1 HEIGHT=1 border=0 SRC="(http://media.preferences.com/ping?
- <ML_SD=IntuitTE_Intuit_1x1_RunOfSite_Any) &db_afcr=4B31-C2FB-10E2C&event=reghome&
- <group=register& time=1999.10.27.20.5 6.37">

Later web bugs (web 2.0) are not included here due to the complexity and length of the code (but see the 3rd-party elements, or '3pes', at http://www.knowyourelements.com/).[5] It is noticeable that this code is extremely opaque and difficult to understand, even for experienced computer programmers. Indeed, one suspects an element of obfuscation, a programming technique to reduce the readability of the code in order to shield the company from observation. So far, in checking a number of web bugs on a variety of websites, I have been unable to find one that supplies any commentary on what exactly the code is doing, beyond a short privacy policy statement. Again Ghostery (2012b) usefully supplies us with some

general information on the web bug, such as the fact that it has been found on over 100,000 websites across the Internet, and that the data collected is 'anonymous (browser type), pseudonymous (IP address)', the data is not shared with third parties but no information is given on their data retention policies. As of 2nd March, 2012, Ghostery reported that it was tracking 829 different web bugs across the Internet. This is a relatively unregulated market in user behavior, tracking and data collection, which currently has a number of self-regulatory bodies, such as the Network Advertising Initative (NAI). As Madrigal reports: 'In essence, [the NAI] argued that users do not have the right to *not* be tracked. "We've long recognized that consumers should be provided a choice about whether data about their likely interests can be used to make their ads more relevant," [they] wrote. "But the NAI code also recognizes that companies sometimes need to continue to collect data for operational reasons that are separate from ad targeting based on a user's online behavior."… Companies "need to continue to collect data," but that contrasts directly with users desire "not to be tracked"' (Madrigal, 2012).

These web bugs, beacons, pixels, and tags, as they are variously called, form part of the dark-net surveillance network that users rarely see, even though it is profoundly changing their experience of the internet in real-time by attempting to second guess, tempt, direct and nudge behavior in particular directions (see Parry, 2011). Ghostery ranked the web bugs in 2010 and identified the following as the

most frequently encountered (above average): Revenue Science (250x), OpenX (254x), AddThis (523.6x), Facebook Connect (529.8x), Omniture (605.7x), Comscore Beacon (659.5x), DoubleClick (924.4x), QuantCast (1042x), Google Adsense (1452x), Google Analytics (3904.5x) (Ghostery, 2011). As can be seen in terms of relative size of encounter, Google is clearly the biggest player by a long distance in the area of user statistics collection. This data is important because, as JP Morgan's Imran Khan explained, a unique visitor to each website at Amazon (e-commerce) is generating $189 per user, at Google (search) it is generating $24 per user, and although Facebook (social networking) is only generating $4 per user, this is a rapidly growing number (Yarrow, 2011). Keeping and holding these visitors, through real-time analytics, customer history, behavioural targeting, etc. is increasingly becomong extremely profitable. Ghostery (2010) has performed a useful analysis of their web bug database that attempts to categorise the web bugs found into 16 different types, which I have re-categorised into five main types: (1) Advertiser/Marketing Services, (2) Analysis/Research Services, (3) Management Platforms, (4) Verification/Privacy Services:

1. **Advertiser/Marketing Services**:

- a. Advertiser: A company sponsoring advertisement and ultimately responsible for the message delivered to the consumer. Example: AT&T

- b. Exchange: A provider of marketplace connecting advertisers to ad networks and data aggregators (online and off), often facilitating multiple connections and bidding processes. Example: Right Media
- c. Network: A broker and often technology provider connecting advertisers and publishers. (web site operators) Example: Burst Media
- d. Publisher: Website operator who displays ads for advertiser(s) in various types of campaigns. Example: The New York Times

2. **Analysis/Research Services**:

- a. Online Data Aggregator: Collects data from online publishers and provides it to advertisers either directly or via exchange. Example: BlueKai
- b. Offline Data Aggregator: Collects data from a range of offline sources and provides data to advertisers directly or via exchange. Experian
- c. Optimizer: Provider of analytics technology and services for ROI assessment and content optimization purposes. Example: ROILabs
- d. Research: Collects data for market research purposes where no ads are serviced through this data. Example: Example: Safecount
- e. Analytics Provider: Provider of cross-platform statistical analysis to understand market effectiveness and audience segmentation. Example: Google Analytics

- f. Retargeter: Providers of technologies that allow publishers to identify their visitor when they place ads on third party sites. Example: Fetchback

3. **Management Platforms**:

- a. Demand-Side Platform: A technology provider that allows marketers to buy inventory across multiple platforms or exchanges. DSPs often layer in custom optimization, audience targeting, real-time bidding and other services. Example: Invite Media
- b. Supply-Side Platform: A technology provider that allows publishers to access advertiser demand across multiple platforms or exchanges. SSPs often layer in custom yield optimization, audience creation, real-time bidding and other services. Example: AdMeld
- c. Ad Server: Technology that delivers and tracks advertisements independently of the web site where the ad is being displayed. Example: DoubleClick DART
- d. Agency: Provider of creative and buying services (both audience and data) for advertisers. Example: MediaCom

4. **Verification/Privacy Services**:

- a. Ad Verification: Certifies or classifies webpages in an effort to prevent advertisers' campaigns from running on unsavory or blocked content, and/or protects advertisers from having other companies run their ads incorrectly. Example: ClickForensics

- b. Online Privacy: Technology providers that deliver information and transparency to consumers on how 3rd party companies gather and use their data. Example: Better Advertising

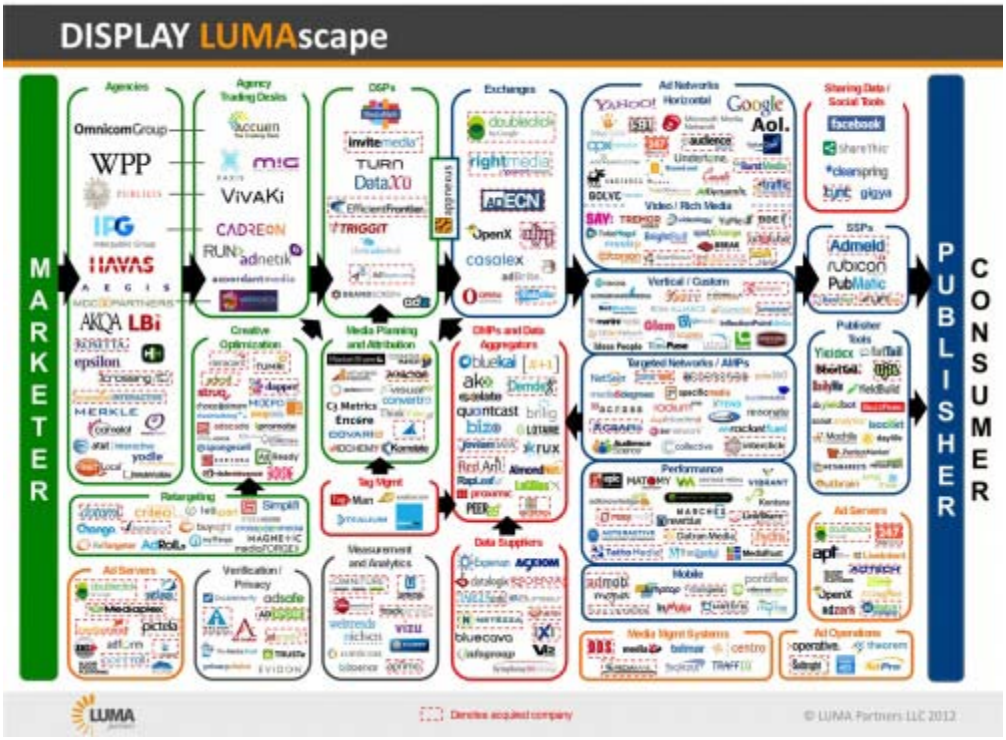


Image 1: Display Advertising Technology Landscape (Luma, 2012)

Ghostery gives a useful explanation of how these companies interoperate to perform a variety of services for advertising and marketing clients:

> A company like Turn Media is a technology provider that allows marketers to buy inventory across multiple platforms or exchanges, or a Demand-Side Platform. They provide services

> for marketers and agencies to centrally manage buying, planning, targeting, and optimizing media opportunities. Reasonably speaking, however, you could also technically classify them as an Optimizer because this process is included under the umbrella of the platform. Turn [Media] is deeply data driven and partners with multiple data providers including BlueKai, TargusInfo, eXelate, and others (Ghostery, 2010).

Of course, one element missing from this typology is that of surveillance, and indeed it is no surprise that web bugs perform part of the tracking technologies used by companies to monitor staff. For example, in 2006 Hewlett Packard used web bugs from readnotify.com to trace insider leaks to the journalist Dawn Kawamoto and later confirmed in testimony to a U.S. House of Representatives subcommittee that it's 'still company practice to use e-mail bugs in certain cases' (Evers, 2006; Fried, 2006).

As can be seen, this is an extremely textured environment that currently offers little in terms of diagnosis or even warnings to the user. The industry itself, which prefers the term "clear GIF" to web bug, is certainly keen to avoid regulation and keeps itself very much to itself in order to avoid raising too much unwarranted attention. Some of the current discussions over the direction of regulation on this issue have focused on the "do not track" flag, which would signal a user's opt-out preference within an HTTP header. Unfortunately, very few companies

respect the "do not track" header and there is currently no legal requirement that they do so in the US, or elsewhere (W3C, 2012). Although one can see in this context the current debate over the EU ePrivacy Directive, where the Article 29 Working Party (A29 WP) has stated that 'voluntary plans drawn up by Europe's digital advertising industry representatives, the European Advertising Standards Alliance (EASA) and IAB Europe, do not meet the consent and information requirements of the recently revised ePrivacy Directive' (Baker, 2012).

One of the newer, and perhaps indicative directions of travel of these new web bugs under development is called PersianStat, which claims to keep 'an eye on 1091622 websites': an Iranian web tracking and data analytics website, it shows that this new code ecology is not purely a Western phenomenon. With the greater use of computational networked devices in everyday life, from mobile phones to GPS systems, these forms of tracking systems will only become more invasive and aggressive in collecting data from our everyday life and encounters. Indeed, it is unsurprising to find that Americans, for example, are not comfortable with the growth in use of these tracker technologies. Pew (2012) found:

> that 73 percent of Americans said they would 'not be okay' with being tracked (because it would be an invasion of privacy)… Only 23 percent said they'd be 'okay' with tracking (because it would lead to better and more personalized search results)…Despite all those

> high-percentage objections to the idea of being tracked, less than half of the people surveyed -- 38 percent -- said they knew of ways to control the data collected about them. (Garber, 2012; Pew, 2012).

This contradiction between the ability of these computational systems and surfaces to supply a commodity to the user, and the need to raise income through the harvesting of data which is in turn sold to advertisers and marketing companies, shows that this is an unstable situation. It also serves to demonstrate the extent to which users are just not aware of the subterranean depths of their computational devices and the ability of these general computing platforms to disconnect the user interface from the actual intentions or functioning of the device, whilst giving the impression to the user that they remain fully in control of the computer. As Garber observes, 'underground network, surface illusion… How much do we actually want to know about this stuff? Do we truly want to understand the intricacies of data-collection and personalization and all the behind-the-screen work that creates the easy, breezy experience of search ... or would we, on some level, prefer that it remain as magic?' (Garber, 2012). An issue helpfully illustrated by the next case study of the Stuxnet virus, which shows the extent to which the magic of software can conceal its true function.

Stuxnet

Stuxnet[6] is a computer worm which experts now believe was aimed at the Iranian uranium-enrichment facility at Natanz, Iran.[7] The Stuxnet worm, a subclass of computer virus, copied itself repeatedly across computer systems until it found the host that met its 'strike conditions', that is, the location it was designed to attack, and activated its 'digital warhead', which may monitor, damage, or even destroy its target. The name, 'Stuxnet,' is 'derived from some of the filename/strings in the malware - mrxcls.sys, mrxnet.sys', the first part, 'stu', comes from the (.stub) file, mrxcls.sys; the second part, 'xnet', comes from mrxnet.sys (Kruszelnicki, 2011; mmpc2, 2010). Due to the sophistication of the programming involved, this worm is considered to have reached a new level in cyberwarfare. Stuxnet has been called the first 'weaponized' computer virus, and it would have required huge resources, like a test facility to model a nuclear plant, to create and launch it (Cherry, 2010). As Liam O Murchu, an operations manager for Symantec, explained:

> Unlike the millions of worms and viruses that turn up on the Internet every year, this one was not trying to steal passwords, identities or money. Stuxnet appeared to be crawling around the world, computer by computer, looking for some sort of industrial operation that was using a specific piece of equipment, a Siemens S7-300 programmable logic controller. (60 Minutes, 2012b)

The Stuxnet worm works by undertaking a very complex stealth infection and covers its tracks by recording data from the nuclear processing system which it then plays back to the operators to disguise that it is actually gently causing the centifuges to fail. This is known as a 'man-in-the-middle attack' because it fakes industrial process control sensor signals so an infected system does not exhibit abnormal behavior and therefore raise alarm. Again, cleverly, the faults it creates in the plant are likely to occur weeks after the sabotaged effort, and in a targeted way, through the fatiguing of the motors – this looks like a standard failure rather than an attack. Indeed, Iran later confirmed that a number of its centrifuges had been affected by an attack (CBS News, 2010). A 'senior Iranian intelligence official said an estimated 16,000 computers were infected by the Stuxnet virus' (Assocated Press, 2012). The Stuxnet worm is also interesting because it has built-in *sunset code* that causes the worm to erase itself after 24 June, 2012, and hence hide its tracks. As Zetter explains:

> once the code infects a system, it searches for the presence of two kinds of frequency converters made by the Iranian firm Fararo Paya and the Finnish company Vacon, making it clear that the code has a precise target in its sights… Stuxnet begins with a nominal frequency of 1,064 Hz… then reduces the frequency for a short while before returning it back to 1,064 Hz… Stuxnet [then] instructs the speed to increase to 1,410 Hz, which is 'very close to the maximum speed the spinning

> aluminum IR-1 rotor can withstand mechanically',… [but] before the rotor reaches the tangential speed at which it would break apart… within 15 minutes after instructing the frequency to increase, Stuxnet returns the frequency to its nominal 1,064 Hz level. Nothing else happens for 27 days, at which point a second attack sequence kicks in that reduces the frequency to 2 Hz, which lasts for 50 minutes before the frequency is restored to 1,064 Hz. Another 27 days pass, and the first attack sequence launches again, increasing the frequency to 1,410 Hz, followed 27 days later by a reduction to 2 Hz. (Zetter, 2011)

Stuxnet disguises all of this activity by overriding the data control systems and sending commands to disable warning and safety controls that would normally alert plant operators to these dangerous frequency changes. Stuxnet is intriguing because it is not a general purpose attack, but designed to unload its digital warheads under specific conditions against a specific threat target. It is also remarkable in the way in which it disengages the interface, the screen for the user, from the underlying logic and performance of the machine.

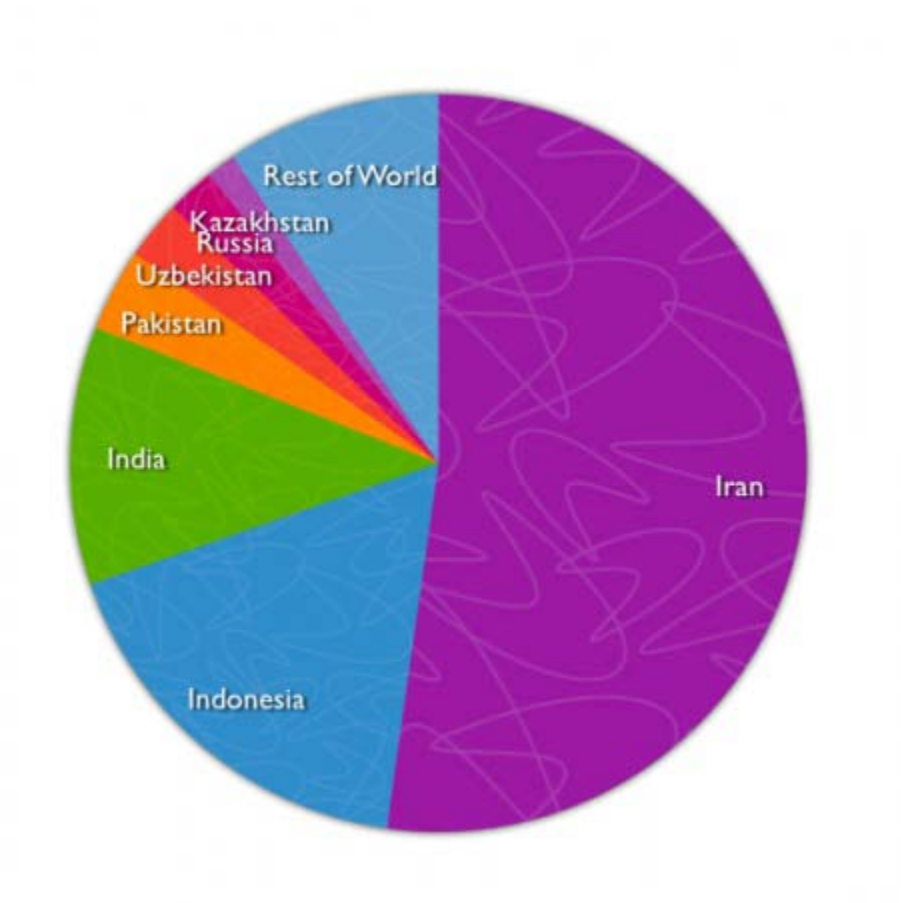
Indeed, there has been a great deal of speculation about whether a state would have been required to develop it due to the complexities involved in being able to test such a worm before releasing it into the wild (Markoff and Sanger, 2010). Richard Clarke, the

former chief of counter-terrorism under Presidents Clinton and Bush, argues that the built-in fail-safes are an important clue to Stuxnet's source and that they point to the kinds of procedures found in a Western government. He says, 'If a [Western] government were going to do something like this…then it would have to go through a bureaucracy, a clearance process, [and] somewhere along the line, lawyers would say, "We have to prevent collateral damage," and the programmers would go back and add features that normally you don't see in the hacks. And there are several of them in Stuxnet' (Gross, 2011). Indeed, the complexities and structure of the worm are such that at least thirty people would have been working on it simultaneously in order to build a worm of this kind (Zetter, 2010). This is especially true of a worm that launched a so-called 'zero-day attack', that is, using a set of techniques that are not public nor known by the developer of the attacked system, in this case Microsoft and Siemens. In actuality it was remarkable for exploiting four different zero-day vulnerabilities (Gross, 2011). Because of the layered approach to its attack and the detailed knowledge required of Microsoft Windows, SCADA (Supervisory Control And Data Acquisition) and PLCs (Programmable Logic Controllers) systems, this would have been a very large project to develop and launch. Indeed, Eric Byres, chief technology officer for Byres Security, has stated: 'we're talking man-months, if not years, of coding to make it work the way it did' (quoted in Zetter, 2010).

The two chief capabilities of Stuxnet are: (1) to identify its target precisely using a number of software based markers that give the physical identity of the location away. Indeed, 'attackers [had] full, and I mean this literally, full tactical knowledge of every damn detail of [the Natanz] plant' (60 Minutes, 2012b); and (2) the capability to disengage control systems from physical systems and to provide a stealth infection into the computer system that would fool the operators of the plant (also known as a 'man-in-the-middle attack'). This was achieved through the use of two 'digital warheads', called 417 and 315. The smaller, 315, was designed to slowly reduce the speed of rotors leading to cracks and failures, and the second larger warhead, 417, manipulated valves in the centrifuge and faking industrial process control sensor signals by modeling the centifuges which were grouped into 164 cascades (Langner, 2011). Indeed, Langner (2011) described this evocatively as 'two shooters from different angles'. The Stuxnet worm was launched some time in 2009/2010 and shortly afterwards:[8]

> the all-important centrifuges at Iran's nuclear fuel enrichment facility at Natanz began failing at a suspicious rate. Iran eventually admitted that computer code created problems for their centrifuges, but downplayed any lasting damage. Computer security experts now agree that code was a sophisticated computer worm dubbed Stuxnet, and that it destroyed more than 1,000 centrifuges. (60 Minutes, 2012a)

The origin of the name Stuxnet is hypothesized from an analysis of the approximately 15,000 lines of programming code. The analysis performed by Langner (2011) and others, was a close reading and reconstruction of the programming logic by taking the machine code, disassembling it and then attempting to convert it into the C programming language. The code could then be analysed for system function calls, timers, and data structures, in order to try to understand what the code was doing (Langner, 2011). Indeed, as part of this process a reference to 'Myrtus' was discovered, and the link made to 'Myrtus as an allusion to the Hebrew word for Esther. The Book of Esther tells the story of a Persian plot against the Jews, who attacked their enemies pre-emptively' (Markoff and Sanger, 2010).[9] Whilst no actor has claimed responsibility for Stuxnet, there is a strong suspicion that either the United States or Israel had to be involved in the creation of such a sophisticated attack virus (see Sanger, 2012). Its attack appears to have been concentrated on a number of selected areas, with Iran at the centre (see table 1).



-
  - Iran - 52.2%
  - Indonesia - 17.4%
  - India - 11.3%
  - Pakistan - 3.6%
  - Uzbekistan- 2.6%
  - Russia - 2.1%
  - Kazakhstan - 1.3%
  - Rest of World - 9.4%

Table 1: Percentage Distribution of Stuxnet Infections by Region (adapted from Matrosov et al n.d.).

Clearly, this kind of attack could be mobilized at targets other than nuclear enrichment facilities, and indeed the stealth and care with which it attempts to fool the operators of the plants shows that computational devices will undoubtedly be targets for monitoring, surveillance, control and so forth in the future. But, of course, once the code for undertaking this kind of sophisticated cyberattack is out in the open it is relatively trivial to decode the computer code and learn techniques that would have taken many years of development in a very short time. As Sean McGurk explains, 'you can download the actual source code of Stuxnet now and you can repurpose it and repackage it and then, you know, point it back towards wherever it came from' (60 Minutes, 2012b). Indeed, a different worm, called Duqu, has already been discovered, albeit with purposes linked to the collection of the data on industrial control systems and structures, a so-called 'Trojan' (Hopkins, 2011).[10] As Alexander Gostev reports:

> There were a number of projects involving programs based on the 'Tilded' [i.e. Stuxnet] platform throughout the period 2007-2011. Stuxnet and Duqu are two of them – there could have been others, which for now remain unknown. The platform continues to develop, which can only mean one thing – we're likely to see more modifications in the future. (2012)

The increased ability of software and code via computational devices to covertly monitor, control and mediate, both positively and negatively, is not just a case of interventions for deceiving the human and non-human actors that make up part of these assemblages. In the next section I want to look at the willing compliance with data collection, indeed the enthusiastic contribution of real-time data to computational systems as part of the notion of lifestreams, and more particularly the quantified self movement.

Lifestreams

Lastly, I want to connect these developments in web-bugs and worms with the growth in the use of self-monitoring technologies called lifestreaming, or the notion of the quantified self.[11] These have expanded in recent years as the 'real-time streams' platforms have expanded, like Twitter and Facebook. Indeed, some argue that 'we're finally in a position where people volunteer information about their specific activities, often their location, who they're with, what they're doing, how they feel about what they're doing, what they're talking about…We've never had data like that before, at least not at that level of granularity' (Rieland, 2012). This has been usefully described by the *Economist*, who argue that the:

> idea of measuring things to chart progress towards a goal is commonplace in large organisations. Governments tot up trade

> figures, hospital waiting times and exam results; companies measure their turnover, profits and inventory. But the use of metrics by individuals is rather less widespread, with the notable exceptions of people who are trying to lose weight or improve their fitness…But some people are doing just these things. They are an eclectic mix of early adopters, fitness freaks, technology evangelists, personal-development junkies, hackers and patients suffering from a wide variety of health problems. What they share is a belief that gathering and analysing data about their everyday activities can help them improve their lives—an approach known as 'self-tracking', 'body hacking' or 'self-quantifying'. (2012)

This phenomena of using computational devices to monitor health signals and to feed them back into calculative interfaces, data visualisations, real-time streams, etc. is the next step in social media. This closes the loop of personal information online, which, although it remains notionally private, is stored and accessed by corporations who wish to use this biodata for data mining and innovation surfacing. For example:

> The Zeo [headband]... has already generated the largest-ever database on sleep stages, which revealed differences between men and women in REM-sleep quantity. Asthmapolis also hopes to

> pool data from thousands of inhalers fitted with its Spiroscout [asthma inhaler] sensor in an effort to improve the management of asthma. And data from the Boozerlyzer [alcohol counting] app is anonymised and aggregated to investigate the variation in people's response to alcohol. (Economist, 2012)

Lifestreams were originally an idea from David Gelernter and Eric Freeman in the 1990s (Freeman, 1997; Gelernter, 2010), which they described as:

> a time-ordered stream of documents that functions as a diary of your electronic life; every document you create and every document other people send you is stored in your lifestream. The tail of your stream contains documents from the past (starting with your electronic birth certificate). Moving away from the tail and toward the present, your stream contains more recent documents -- papers in progress or new electronic mail; other documents (pictures, correspondence, bills, movies, voice mail, software) are stored in between. Moving beyond the present and into the future, the stream contains documents you *will* need: reminders, calendar items, to-do lists. You manage your lifestream through a small number of powerful operators that allow you to transparently store information, organize information on demand, filter and monitor incoming information, create reminders and calendar items in an integrated fashion, and 'compress' large numbers of

> documents into overviews or executive summaries. (Freeman, 2000)

Gelernter originally described these ‘chronicle streams’ (Gelernter, 1994), highlighting both their narrative and temporal dimensions related to the storage of documentation and texts. Today we are more likely to think of them as ‘real-time streams’ and the timeline functions offered by systems like Twitter, Facebook and Google+. These are increasingly the model of interface design that is driving the innovation in computation, especially in mobile and locative technologies. However, in contrast to the document-centric model that Gelernter and Freeman were describing, there are also the micro-streams of short updates, epitomized by Twitter, which has short text-message sized 140 character updates. Nonetheless this is still enough text space to incorporate a surprising amount of data, particularly when geo, image, weblinks, and so forth are factored in. Stephen Wolfram was one of the first people to collect their data systematically. As he explains, Wolfram started in 1989: 'So email is one kind of data I’ve systematically archived. And there’s a huge amount that can be learned from that. Another kind of data that I’ve been collecting is keystrokes. For many years, I’ve captured every keystroke I’ve typed—now more than 100 million of them' (Wolfram, 2012).

This kind of self-collection of data is certainly becoming more prevalent and in the context of reflexivity and self-knowledge, it raises interesting

questions about the increasing use of mathematics and computation to understand and control the self. The scale of data that is collected can also be relatively large and unstructured. Nonetheless, better data management and techniques for searching and surfacing information from unstructured or semi-structured data will no doubt be revealing about our everyday patterns in the future.[12]

Mobile 'apps' - small, relatively contained applications that usually perform a single specific function - have accelerated this way of collecting and sending data. For example, the Twitter app on the iPhone allows the user to send updates to their timeline, but also search other timelines, check out profiles, streams and so on. When created as apps, however, they are also able to use the power of the local device, especially if it contains the kinds of sophisticated sensory circuitry that is common in smartphones, to log GPS geographic location, direction, etc. This is when life-streaming becomes increasingly similar to the activity of web bugs in monitoring and collecting data on the users that are active on the network. Indeed, activity streams have become a standard which is increasingly being incorporated into software across a number of media and software practices (see ActivityStreams n.d.). An activity stream essentially encodes a user event or activity into a form that can be computationally transmitted and later aggregated, searched and processed:

- In its simplest form, an activity consists of an *actor*, a *verb*, an *object*, and a *target*. It tells the story of a person performing an action on or with an object -- 'Geraldine posted a photo to her album' or 'John shared a video'. In most cases these components will be explicit, but they may also be implied. (ActivityStreamsWG, 2011, original emphasis)

  This data and activity collection is only part of the picture, however. In order to become reflexive data it must be computationally processed from its raw state, which may be structured, unstructured, or a combination of the two. At this point it is common for the data to be visualized, usually through a graph or timeline, but there are also techniques such as heat-maps, graph theory, and so forth that enable the data to be processed and reprocessed to tease out patterns in the underlying data set. In both the individual and aggregative use case, in other words for the individual user (or lifestreamer) or organization (such as Facebook), the key is to pattern match and compare details of the data, such as against a norm, a historical data set, or against a population, group, or class or others.[13]

  The patterned usage is therefore a dynamic real-time feedback mechanism, in terms of providing steers for behaviour, norms and so forth, but also offering a documentary narcissism that appears to give the user an existential confirmation and status. Even in its so-called gamification forms, the awarding of

competitive points, badges, honours and positional goods more generally is the construction of a hierarchical social structure within the group of users. It also encourages the user to think of themselves as a set of partial objects, fragmented dividuals, or loosely connected properties, collected as a time-series of data-points and subject to intervention and control. This can be thought of as a computational 'care of the self', facilitated by an army of oligopticans (Latour, 2005) in the wider computational environment that observe and store behavioural and affective data. However, this self is reconciled through the code and software that makes the data make sense. The code and software are therefore responsible for creating and maintaining the meaning and narratives through a stabilisation and web of meaning for the actor.[14]

I now want to turn to how we might draw these case studies together to think about living in code and software and the implications for wider study in terms of research and theorisation of computational society.

Conclusions

It seems that a thread runs through web bugs, viruses and now life-streaming itself: this is data collection, monitoring and real-time feedback, whether overt or covert. Whilst we can continue to study these phenomena in isolation, and indeed there can be very productive knowledge generated from this kind of research, it seems to me that we

need to attend to the computationality represented in code and software to better understand such software ecologies (Berry, 2011).

One of the most interesting aspects of these systems is that humans in many cases become the vectors that both enable the data transfers and carry the data that fuels the computational economy. Our movements between systems, carrying USB sticks and logging into email accounts and distant networks, creates the channels through which data flows or an infection is spread. The ability of these viruses to take on some of the features of web bugs and learn our habits and preferences in real-time whilst secreting themselves within our computer systems raises important questions, particularly in relation to the complexity and obfuscated nature of the code and its ability to track and collect data surreptitiously. However, users are actively downloading apps that advertise the fact that they collect this data and seem to genuinely find an existential relief or recognition in their movements being recorded and available for later playback or analysis. Web bugs are in many ways life streams - albeit life streams that have not been authorized by the user whom they are monitoring. This collection of what we might call *compactants* are designed to *passive-aggressively* record data.[15] With the notion of *compactants* (computational actants) I want to particularly draw attention to this passive-aggressive feature of computational agents that are collecting information. Both in terms of their passive quality – under the surface, relatively benign and

silent – but also the fact that they are aggressive in their hoarding of data – monitoring behavioural signals, streams of affectivity and so forth. The word *compact* also has useful overtones of having all the necessary components or functions neatly fitted into a small package, and compact as in conciseness in expression. The etymology from the Latin *compact* for closely put together, or joined together, also neatly expresses the sense of what web bugs and related technologies are. That is, compactants are interesting in terms of the distributed agency they enable, which can be understood through the notion of *companion actants* (see Haraway, 2003).

Interestingly, compactants are structured in such a way that they can be understood as having a dichotomous structure of data-collection/visualisation, each of which is a specific mode of operation. Naturally, due to the huge quantities of data that is often generated, the computational processing and aggregation is often offloaded to the 'cloud', or server computers designed specifically for the task and accessed via networks. Indeed, many viruses, for example, often seek to 'call home' to report their status, upload data, or offer the chance of being updated, perhaps to a more aggressive version of themselves or to correct bugs.

We might also think about the addressee of these wider computational systems made up of arrays or networks of compactants, which in many cases is a future actor. Within the quantified-self movement there is an explicit recognition that the 'future self'

will be required to undo bad habits and behaviours of the present-self. That is, that there is an explicit normative context to a *future* self, who you, as the *present* self may be treating unfairly, immorally or without due regard to what has been described as 'future self continuity' (Tugend, 2012). This inbuilt tendency toward the *futural* is a fascinating reflection of the internal temporal representation of time within computational systems, that is time-series structured streams of real-time data, often organised as lists. Therefore the past (as stored data), present (as current data collection, or processed archival data), and future (as both the ethical addressee of the system and potential provider of data and usage) are often deeply embedded in the code that runs these systems. In some cases the future also has an objective existence as a probabilistic projection, literally a *code-object*, which is updated in real-time and which contains the major features of the future state represented as a model; computational weather prediction systems and climate change models are both examples of this.

There are many examples of how attending to the code and software that structures many of the life, memory and biopolitical systems and industries of contemporary society could yield similarly revealing insights into both our usage of code and software, but also the structuring assumptions, conditions and affordances that are generated. Our use of computational models is growing, and our tendency is to confuse the screenic representation visualised by code/software with what we might call the real –

not to mention our failure to appreciate the ways in which code's mediation is co-constructive of, and deeply involved in, the stabilisation of everyday life today. Even so, within institutional contexts, code/software has not fully been incorporated into the specific logics of these social systems, and in many ways undermines these structural and institutional forms. We must remain attentive to the fact that software engineering itself is a relatively recent discipline and its efforts at systematisation and rationalisation are piecemeal and incomplete, as the many hugely expensive software system failures attests. Of course, this code/software research is not easy, the techniques needed are still in their infancy, and whilst drawing on a wide range of scholarly work from the sciences, social sciences and the arts and humanities we are still developing our understanding. But this should give hope and direction to the critical theorists, both of the present looking to provide critique and counterfactuals, but also *of* the future, as code/software is a particularly rich site for intervention, contestation and the *unbuilding* of code/software systems.[16]

Acknowledgements

I am very grateful to the *Forskningsrådet* (Research Council of Norway) for the *Yggdrasil* fellowship ref: 211106 which funded my sabbatical in Oslo in 2012. I would also like to thank Anders Fagerjord, *Institutt for medier og kommunikasjon* (IMK), University of Oslo, for the kind invitation to be based at the

university. An earlier version of this chapter was presented at UnlikeUs in March 2012, at the University of Amsterdam, and I would like to thank Geert Lovink for the kind invitation to present this work. I am also grateful to have had the opportunity to present versions of the chapter in this book to: PhiSci seminar series, organised by Rani Lill Anjum, CauSci (Causation in Science) and the UMB School of Economics and Business; *Institutt for medier og kommunikasjon* (IMK) seminar series, invited by Espen Ytreberg, University of Oslo; Digital Humanities Workshop, organized by Caroline Bassett, University of Sussex; the Media Innovations Colloquium organized by Tanja Storsul, *Institutt for medier og kommunikasjon* (IMK), University of Oslo; and the Archive in Motion workshop, *Nasjonal Bibliotek* organised by Ina Blom, University of Oslo. I would also like to express my deepest thanks to Michael Najjar for the kind permission to use his work, 'The sublime brain [of Jonathon]', for the cover of the book which represents a neuronal frontal portait of an individual, for more of his excellent work please see [1]. Many thanks are also due to Trine for proofing the documents included in this living book.

Bibliography

60 Minutes (2012a) 'Fmr. CIA head calls Stuxnet virus "good idea"', *60 Minutes*, accessed 04/03/2012, http://www.cbsnews.com/8301-18560_162-57388982/fmr-cia-head-calls-stuxnet-virus-good-idea/

60 Minutes (2012b) 'Stuxnet: Computer worm opens new era of warfare', *60 Minutes*, accessed 04/03/2012, http://www.cbsnews.com/8301-18560_162-57390124/stuxnet-computer-worm-opens-new-era-of-warfare/

ActivityStreams (n.d.) 'Activity Streams', accessed 04/03/2012, http://activitystrea.ms/

ActivityStreamsWG (2011) 'JSON Activity Streams 1.0', Activity Streams Working Group, accessed 04/03/2012, http://activitystrea.ms/specs/json/1.0/

Associated Press (2012) 'Iran says Stuxnet virus infected 16,000 computers', *Associated Press*, accessed 04/03/2012, http://www.foxnews.com/world/2012/02/18/iran-says-stuxnet-virus-infected-16000-computers/

Berry, D. M. (2011) *The Philosophy of Software: Code and Mediation in the Digital Age*, London: Palgrave.

Baker, J. (2012) 'European Watchdog Pushes for Do Not Track Protocol', accessed 10/03/2012, http://www.pcworld.com/businesscenter/article/25137

3/european_watchdog_pushes_for_do_not_track_protocol.html

CBS News (2010) 'Iran Confirms Stuxnet Worm Halted Centrifuges', *CBSNews*, accessed 04/03/2012, http://www.cbsnews.com/stories/2010/11/29/world/main7100197.shtml

Cherry, S. (2010) 'How Stuxnet Is Rewriting the Cyberterrorism Playbook', *IEEE Spectrum: Inside Technology*, accessed 04/03/2012, http://spectrum.ieee.org/podcast/telecom/security/how-stuxnet-is-rewriting-the-cyberterrorism-playbook

Cryptome (2010) 'Stuxnet Myrtus or MyRTUs?', accessed 04/03/2012, http://cryptome.org/0002/myrtus-v-myRTUs.htm

Deuze, M., Blank, P. and Speers, L. (2012) 'A Life Lived in Media', *Digital Humanities Quarterly*, Winter 2012, Volume 6 Number 1, accessed 29/02/2012, http://digitalhumanities.org/dhq/vol/6/1/000110/000110.html

Dobias, J. (2010) 'Privacy Effects of Web Bugs Amplified by Web 2.0', in Fischer-Hübner, S., Duquenoy, P., Hansen, M., Leenes, R., and Zhang, G. (eds.) *Privacy and Identity Management for Life*, London: Springer.

Economist (2012) 'Counting every moment', *The Economist*, accessed 02/03/2012, http://www.economist.com/node/21548493

EFF (1999) 'The Web Bug FAQ', accessed 02/03/2012, http://w2.eff.org/Privacy/Marketing/

Evans, S. (2012) 'Duqu Trojan used 'unknown' programming language: Kaspersky', CBR Software Malware, accessed 09/03/2012, http://malware.cbronline.com/news/duqu-trojan-used-unknown-programming-language-kaspersky-070312

Evers, J. (2006) 'How HP bugged e-mail', accessed 02/03/2012, http://news.cnet.com/How-HP-bugged-e-mail/2100-1029_3-6121048.html

Eyal, N. (2012) 'How To Manufacture Desire', *TechCrunch*,accessed 05/03/2012,

http://techcrunch.com/2012/03/04/how-to-manufacture-desire/

Frabetti, F. (2010) 'Critical Code Studies', accessed 15/03/2012, http://vimeo.com/16263212

Fried, I. (2006) 'Dunn grilled by Congress', accessed 02/03/2012, http://news.cnet.com/Dunn-grilled-by-Congress/2100-1014_3-6120625.html

Freeman, E. T. (1997) 'The Lifestreams Software Architecture', Ph.D. Dissertation, Yale University Department of Computer Science, May 1997, accessed 02/03/2012, http://www.cs.yale.edu/homes/freeman/dissertation/etf.pdf

Freeman, E. T. (2000) 'Welcome to the Yale Lifestreams homepage!', accessed 02/03/2012, http://cs-www.cs.yale.edu/homes/freeman/lifestreams.html

Garber, M. (2012) 'Americans Love Google! Americans Hate Google!', *The Atlantic*, accessed 02/03/2012, http://m.theatlantic.com/technology/archive/2012/03/

americans-love-google-americans-hate-google/254253/

Gelernter, D. (1994) 'The cyber-road not taken.', *The Washington Post*, April 1994.

Gelernter, D. (2010) 'Time To Start Taking The Internet Seriously', *The Edge*, accessed 02/03/2012, http://www.edge.org/3rd_culture/gelernter10/gelernter10_index.html

Ghostery (2010) 'The Many Data Hats a Company can Wear', accessed 02/03/2012, http://purplebox.ghostery.com/?p=948639073

Ghostery (2011) 'Ghostrank Planetary System', accessed 02/03/2012, http://purplebox.ghostery.com/?p=1016021670

Ghostery (2012a) 'About Ghostery', accessed 02/03/2012, http://www.ghostery.com/about)

Ghostery (2012b) 'About ChartBeat', accessed 02/03/2012, http://www.ghostery.com/apps/chartbeat

Gostev, A. (2012) 'Stuxnet/Duqu: The Evolution of Drivers', SecureList, accessed 02/03/2012, https://www.securelist.com/en/analysis/204792208/Stuxnet_Duqu_The_Evolution_of_Drivers

Gross, M. J. (2011) 'A Declaration of Cyber-War', *Vanity Fair*, accessed 02/03/2012, http://www.vanityfair.com/culture/features/2011/04/stuxnet-201104

Hall, G. (2011) *Digitize Me, Visualize Me, Search Me*, Open Humanities Press, accessed 02/03/2012, http://www.livingbooksaboutlife.org/books/Digitize_Me,_Visualize_Me,_Search_Me

Haraway, D. (2003) *The Companion Species Manifesto: Dogs, People, and Significant Otherness*, Prickly Paradigm Press.

Hayles, N. K. (2004) 'Print Is Flat, Code Is Deep: The Importance of Media-Specific Analysis', *Poetics Today*, 25:1, pp 67-90.

Hopkins, N. (2011) *New Stuxnet' worm targets companies in Europe',* The Guardian, *http://www.guardian.co.uk/technology/2011/oct/19/stuxnet-worm-europe-duqu*

Kruszelnicki, K. (2011) 'Stuxnet opens cracks in Iran nuclear program', accessed 02/03/2012, http://www.abc.net.au/science/articles/2011/10/26/3348123.htm

Langner, R. (2011) 'Ralph Langner: Cracking Stuxnet, a 21st-century cyberweapon', accessed 02/03/2012, http://www.youtube.com/watch?feature=player_embedded&v=CS01Hmjv1pQ

Luma (2012) 'Display Advertising Technology Landscape', accessed 02/06/2012, http://www.lumapartners.com/resource-center/

Madrigal, A. (2012) 'I'm Being Followed: How Google—and 104 Other Companies—Are Tracking Me on the Web', *The Atlantic*, accessed 02/03/2012, http://m.theatlantic.com/technology/archive/2012/02/im-being-followed-how-google-and-104-other-companies-are-tracking-me-on-the-web/253758/

Markoff, J. and Sanger, D. S. (2010) 'In a Computer Worm, a Possible Biblical Clue', *The New York Times*, accessed 04/03/2012, http://www.nytimes.com/2010/09/30/world/middleeast/30worm.html?_r=1

Matrosov, A., Rodionov, E., Harley, D. and Malcho, J. (n.d.) 'Stuxnet Under the Microscope', accessed 04/03/2012, http://go.eset.com/us/resources/white-papers/Stuxnet_Under_the_Microscope.pdf

Mitcham, C. (1998) 'The Importance of Philosophy to Engineering', *Teorema*, Vol. XVII/3, pp. 27-47.

Mittal, S. (2010) 'User Privacy and the Evolution of Third-party Tracking Mechanisms on the World Wide Web', Thesis, accessed 04/03/2012, http://www.stanford.edu/~sonalm/Mittal_Thesis.pdf

Mmpc2 (2010) 'The Stuxnet Sting', accessed 04/03/2012, http://blogs.technet.com/b/mmpc/archive/2010/07/16/the-stuxnet-sting.aspx

Parikka, J. (2012) *Medianatures: The Materiality of Information Technology and Electronic Waste*, Open Humanities Press, accessed 01/06/2012, http://www.livingbooksaboutlife.org/books/Medianatures

Parry, D. (2011) *Ubiquitous Surveillance*, Open Humanities Press, http://www.livingbooksaboutlife.org/books/Ubiquitous_Surveillance

Peterson, D. G. (2012) 'Langner's Stuxnet Deep Dive S4 Video', accessed 04/03/2012, http://www.digitalbond.com/2012/01/31/langners-stuxnet-deep-dive-s4-video/

Pew (2012) 'Search Engine Use 2012', accessed 09/03/2012, http://pewinternet.org/Reports/2012/Search-Engine-Use-2012/Summary-of-findings.aspx

Rieland, R. (2012) 'So What Do We Do With All This Data?', _The Smithsonian_, accessed 04/03/2012, http://blogs.smithsonianmag.com/ideas/2012/01/so-what-do-we-do-with-all-this-data/

Sanger, D. E. (2012) Obama Order Sped Up Wave of Cyberattacks Against Iran, The New York Times, June 1, 2012, accessed 02/06/2012, http://www.nytimes.com/2012/06/01/world/middleeast/obama-ordered-wave-of-cyberattacks-against-iran.html?_r=1

Sense (2012) 'Feel. Act. Make sense', accessed 04/03/2012, http://open.sen.se/

Tugend, A. (2012) 'Bad Habits? My Future Self Will Deal With That', accessed 04/03/2012, http://www.nytimes.com/2012/02/25/business/another-theory-on-why-bad-habits-are-hard-to-break-shortcuts.html?_r=3&pagewanted=all

Wauters, R (2012) '427 million Europeans are now online, 37% uses more than one device: IAB', The Next Web, accessed 01/06/2012, http://thenextweb.com/eu/2012/05/31/427-million-europeans-are-now-online-37-uses-more-than-one-device-iab/

W3C (2012) 'Tracking Protection Working Group', accessed 14/03/2012, http://www.w3.org/2011/tracking-protection/

Wolfram, S. (2012) 'The Personal Analytics of My Life', accessed 09/03/2012, http://blog.stephenwolfram.com/2012/03/the-personal-analytics-of-my-life/

Yarrow, J. (2011) 'CHART OF THE DAY: Here's How Much A Unique Visitor Is Worth', *Business Insider*, accessed 02/03/2012, http://www.businessinsider.com/chart-of-the-day-revenue-per-unique-visitor-2011-1

Zetter, K. (2010) 'Blockbuster Worm Aimed for Infrastructure', But No Proof Iran Nukes Were Target, *Wired*, accessed 02/03/2012, http://www.wired.com/threatlevel/2010/09/stuxnet/

Zetter, K. (2011) 'Report Strengthens Suspicions That Stuxnet Sabotaged Iran's Nuclear Plant', *Wired*, accessed 02/03/2012, http://www.wired.com/threatlevel/2010/12/isis-report-on-stuxnet/

Notes

[1] These include HTTP cookies and Locally Stored Objects (LSOs) and document object model storage (DOM Storage)

[2] 'Cookies are small pieces of text that servers can set and read from a client computer in order to register its "state." They have strictly specified structures and can contain no more than 4 KB of data each. When a user navigates to a particular domain, the domain may call a script to set a cookie on the user's machine. The browser will send this cookie in all subsequent communication between the client and the server until the cookie expires or is reset by the server' (Mittal, 2010: 10).

[3] Ghostery describes itself on its help page: 'Be a web detective. Ghostery is your window into the invisible web – tags, web bugs, pixels and beacons that are included on web pages in order to get an idea of your online behavior. Ghostery tracks the trackers and gives you a roll-call of the ad networks, behavioral data providers, web publishers, and other companies interested in your activity' (Ghostery, 2012a).

[4] For an example see, http://static.chartbeat.com/js/chartbeat.js

[5] Also see examples at: (1) Chartbeat ; (2) Google Analytics ; (3) Omniture ; (4) Advertising.com

[6] A computer worm is technically similar in design to a virus and is therefore considered to be a sub-class of a virus. Indeed, worms spread from computer to computer, often across networks, but unlike a virus, a worm has the ability to transfer itself without requiring any human action. A worm is able to do this by taking advantage of the file or information transport features, such as the networking setup, on a computer, which it exploits to enable it to travel from computer to computer unaided.

[7] One of the ways in which the Stuxnet attack target was identified was through a close reading of the computer code that was disassembled from the worm and the careful analysis of the internal data structures and finite state machine used to structure the attack. Ironically, this was then matched by Ralph Langner with photographs that had been uploaded to the website of the President of Iran, Mahmoud Ahmadinejad, and confirmed the importance of the cascade structure, centrifuge layout and the enriching process by careful analysis of the accidental photographing of background images on computers used by the president see http://www.president.ir/en/9172 (see Peterson, 2012).

[8] The timestamp in the file ~wtr4141.tmp indicates that the date of compilation was on 03/02/2010 (Matrosov et al., n.d.). Although there is suspicion that there may be three versions of the Stuxnet code in response to its discovery: 'Most curious, there were two major variants of the worm. The earliest versions of it, which appear to have been released in

the summer of 2009, were extremely sophisticated in some ways but fairly primitive in others, compared with the newer version, which seems to have first circulated in March 2010. A third variant, containing minor improvements, appeared in April. In Schouwenberg's view, this may mean that the authors thought Stuxnet wasn't moving fast enough, or had not hit its target, so they created a more aggressive delivery mechanism. The authors, he thinks, weighed the risk of discovery against the risk of a mission failure and chose the former' (Gross, 2011).

[9] Although there are some criticisms that this link may be spurious. For instance, Cryptome (2010) argues: It may be that the 'myrtus' string from the recovered Stuxnet file path "b:\myrtus\src\objfre_w2k_x86\i386\guava.pdb" stands for "My-RTUs"as in Remote Terminal Unit.

[10] After having performed detailed analysis of the Duqu code, Kaspersky Labs stated that they 'are 100% confident that the Duqu Framework was not programmed with Visual C++. It is possible that its authors used an in-house framework to generate intermediary C code, or they used another completely different programming language' (Evans, 2012).

[11] See http://quantifiedself.com/

[12] Wolfram further writes: 'It's amazing how much it's possible to figure out by analyzing the various kinds of data I've kept. And in fact, there are many additional kinds of data I haven't even touched on in

this post. I've also got years of curated medical test data (as well as my not-yet-very-useful complete genome), GPS location tracks, room-by-room motion sensor data, endless corporate records—and much much more…And as I think about it all, I suppose my greatest regret is that I did not start collecting more data earlier. I have some backups of my computer filesystems going back to 1980. And if I look at the 1.7 million files in my current filesystem, there's a kind of archeology one can do, looking at files that haven't been modified for a long time (the earliest is dated June 29, 1980)' (2012).

[13] Some examples of visualization software for this kind of life-streaming quantification and visualization are shown on these pages from the Quantified Self website: http://quantifiedself.com/2011/03/personal-data-visualization/ , http://quantifiedself.com/2010/05/jaw-dropping-infographics-for/ , http://quantifiedself.com/2010/05/the-visualization-zoo/ , http://quantifiedself.com/2009/09/visualization-inspiration/

[14] See http://open.sen.se/ for a particularly good example of this: 'Make your data history meaningful. Privately store your flows of information and use rich visualizations and mashup tools to understand what's going on' (Sense, 2012).

[15] Computational actants, drawing the notion of actant from actor-network theory. I also like the association with companion actants, similar in idea to companion species.

[16] Here I tentatively raise the suggestion that a future critical theory of code and software is committed to *un-building*, *dis-assembling*, and *de-formation* of existing code/software systems, together with a necessary intervention in terms of a positive moment in the formation and composition of future and alternative systems.

Readings

Thinking Software

Eric W. Weisstein
What is a Turing Machine?

David Barker-Plummer
Turing Machines

Achim Jung
A Short Introduction to the Lambda Calculus

Luciana Parisi & Stamatia Portanova
Soft Thought (in architecture and choreography)

David M. Berry
Understanding Digital Humanities

Edsger W. Dijkstra
Go To Statement Considered Harmful

Alan M. Turing
Computing Machinery and Intelligence

Martin Gardner
The Fantastic Combinations of John Conway's New Solitaire Game 'Life'

David Golumbia
Computation, Gender, and Human Thinking

Alan M. Turing
Extract from On Computable Numbers, with an Application to the Entscheidungs Problem

Video of a Turing Machine - Overview
Kevin Slavin
How Algorithms Shape Our World
Video shows how these complex computer programs determine: espionage tactics, stock prices, movie scripts, and architecture.

Code Literacy ('iteracy')

David M. Berry
Iteracy: Reading, Writing and Running Code

Ian Bogost
Procedural Literacy: Problem Solving with Programming, Systems, & Play

Cathy Davidson
Why We Need a 4th R: Reading, wRiting, aRithmetic, algoRithms

Jeannette M. Wing
Computational Thinking

Stephan Ramsay
On Building

Edsger W. Dijkstra
On the Cruelty of Really Teaching Computing Science

Louis McCallum and Davy Smith
Show Us Your Screens

A short documentary about live coding practise by Louis McCallum and Davy Smith.

Jeannette M. Wing
Computational Thinking and Thinking About Computing'
Wing argues that computational thinking will be a fundamental skill used by everyone in the world. To reading, writing, and arithmetic, she adds computational thinking to everyones' analytical ability.
why the lucky stiff

Hackety Hack: Learning to Code
why the lucky stiff (or _why) is a computer programmer, talking about learning to code.

Decoding Code

David M. Berry
A Contribution Towards a Grammar of Code

Mark C. Marino
Critical Code Studies

Lev Manovich
Software Takes Command

Dennis G. Jerz
Somewhere Nearby is Colossal Cave: Examining Will Crowther's Original "Adventure" in Code and in Kentucky

Aleksandr Matrosov, Eugene Rodionov, David Harley, and Juraj Malcho, J.
Stuxnet Under the Microscope

Ralph Langner
Cracking Stuxnet, a 21st-century Cyber Weapon
A fascinating look inside cyber-forensics and the processes of reading code to understand how it works and what it attacks.

Stephen Ramsay
Algorithms are Thoughts, Chainsaws are Tools
A short film on livecoding presented as part of the Critical Code Studies Working Group, March 2010, by Stephen Ramsay. Presents a "live reading" of a performance by composer Andrew Sorensen.

Wendy Chun
Critical Code Studies
Wendy Chun giving a lecture on code studies and reading source code.

Federica Frabetti
Critical Code Studies
Federica Frabetti giving a lecture on code studies and reading source code.

David M. Berry
Thinking Software: Realtime Streams and Knowledge in the Digital Age
As software/code increasingly structures the contemporary world, curiously, it also withdraws, and becomes harder and harder for us to focus on as it is embedded, hidden, off-shored or merely forgotten about. The challenge is to bring software/code back into visibility so that we can pay attention to both what it is (ontology/medium), where it has come from (media archaeology/genealogy) but also what it is doing

(through a form of mechanology), so we can understand this 'dynamic of organized inorganic matter'.

Software Ecologies

Gabriella Coleman
The Anthropology of Hackers

Felix Guattari
The Three Ecologies

Robert Kitchin
The Programmable City

Bruno Latour
The Whole is Always Smaller Than Its Parts- A Digital Test of Gabriel Tarde's Monads

Mathew Fuller and Sonia Matos
Feral Computing: From Ubiquitous Calculation to Wild Interactions

Jussi Parikka
Media Ecologies and Imaginary Media: Transversal Expansions, Contractions, and Foldings

David Gelernter
Time to Start Taking the Internet Seriously

Adrian Mackenzie
The Problem of Computer Code: Leviathan or Common Power?

Adrian Mackenzie
Wirelessness as Experience of Transition

Thomas Goetz
Harnessing the Power of Feedback Loops

Christian Ulrik Andersen & Søren Pold
The Scripted Spaces of Urban Ubiquitous Computing: The Experience, Poetics, and Politics of Public Scripted Space

B.J. Fogg, Gregory Cuellar, and David Danielson
Motivating, Influencing, and Persuading Users

Alexander R. Galloway
"Deleuze and Computers" - Alexander R. Galloway
"Deleuze and Computers" - a lecture by Alexander R. Galloway at the W.E.B. Du Bois Library at the University of Massachusetts Amherst on December 2nd, 2011.

Gary Wolf
The Quantified Self
The notion of using computational devices in everyday life to record everything about you.

Gary Kovacs
Tracking the Trackers
As you surf the Web, information is being collected about you.

Michael Najjar
How Art Envisions Our Future
Data, information, computation, and technology mediated through art

Attributions

Andersen, C. U. and Pold, S. (2011) 'The Scripted Spaces of Urban Ubiquitous Computing: The Experience, Poetics, and Politics of Public Scripted Space', *Fibreculture*, 19.
Link: http://nineteen.fibreculturejournal.org/fcj-133-the-scripted-spaces-of-urban-ubiquitous-computing-the-experience-poetics-and-politics-of-public-scripted-space/

Berry, D. M. (2008) 'A Contribution Towards a Grammar of Code', *Fibreculture*, 13.
Link: http://thirteen.fibreculturejournal.org/fcj-086-a-contribution-towards-a-grammar-of-code/

Berry, D. M. (2011) 'Iteracy: Reading, Writing and Running Code'.
Link: http://stunlaw.blogspot.com/2011/09/iteracy-reading-writing-and-running.html

Berry, D. M. (2012) *Understanding Digital Humanities*, London: Palgrave Macmillan.
Link: http://www.palgrave.com/PDFs/9780230292642.pdf
Licence © 2012 David M. Berry, Palgrave Macmillan. Sample chapter available for open access.

Berry, D. M. (2012) 'Thinking Software: Realtime Streams and Knowledge in the Digital Age'.
Link to Vimeo: http://vimeo.com/39256099

Bogost, I. (2005) 'Procedural Literacy: Problem Solving with Programming, Systems, & Play, Telemedium'.
Link:
http://www.bogost.com/downloads/I.%20Bogost%20Procedural%20Literacy.pdf
Licence: © 2005 Telemedium. Made freely available on author's website.

Coleman, G. (2010) 'The Anthropology of Hackers,' *The Atlantic.*
Link:
http://www.theatlantic.com/technology/archive/2010/09/the-anthropology-of-hackers/63308/
Licence: © 2010 The Atlantic Monthly Group. All Rights Reserved. Made available via a link to *The Atlantic.*

Chun, W. (2010) 'Critical Code Studies'.
Link to Vimeo: http://vimeo.com/16328263

Davidson, C. (2012) 'Why We Need a 4th R: Reading, wRiting, aRithmetic, algoRithms,' *DML Central.*

Link: http://dmlcentral.net/blog/cathy-davidson/why-we-need-4th-r-reading-writing-arithmetic-algorithms
Licence: ©2012 Davidson*, Creative Commons Attribution 3.0 License*. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Dijkstra, E. W. (1998) 'On the Cruelty of Really Teaching Computing Science'.
Link: http://virtual.itca.edu.sv/dokeos/sinapsis/cd/doctos-sw-libre/docus-ewd/EWD1036%20-%20On%20the%20cruelty%20of%20really%20teaching%20computing%20scienc.pdf
Licence: © 1998 Dijkstra. Made available here via transcription.

Gardner, M. (1970) 'The Fantastic Combinations of John Conway's New Solitaire Game "Life"', *Scientific American*, 223 (October): 120-123.
Link:
http://www.ibiblio.org/lifepatterns/october1970.html
Licence: ©1970 Scientific American. Made available via a link to ibiblio.

Galloway, A. R. (2003) '"Deleuze and Computers" - Alexander R. Galloway.'
Link to Youtube:
http://www.youtube.com/watch?v=fBZPJNoJWHk

Golumbia, D. (2003) 'Computation, Gender, and Human Thinking,' *Differences: A Journal of Feminist Cultural Studies,* 14 (2), Summer: 27-48.

Guattari, F. (1989) 'The Three Ecologies,' *New Formations*, 8.
Link: http://www.amielandmelburn.org.uk/collections/newformations/08_131.pdf

Fogg, B. J., Cuellar, G., and Danielson, D. (2003) 'Motivating, Influencing, and Persuading Users,' In Jacko, J. and Sears A. (eds.), *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications.*
Link: http://bjfogg.com/hci.pdf

Fuller, M. and Matos, S. (2011) 'Feral Computing: From Ubiquitous Calculation to Wild Interactions,' *Fibreculture*, 19.
Link: http://nineteen.fibreculturejournal.org/fcj-135-feral-computing-from-ubiquitous-calculation-to-wild-interactions/

medium, provided the original author and source are credited.

Frabetti, F. (2010) 'Critical Code Studies'.
Link to Vimeo: http://vimeo.com/16263212

Goetz, T. (2011) 'Harnessing the Power of Feedback Loops,' *Wired.*
Link:
http://www.wired.com/magazine/2011/06/ff_feedbackloop/
Licence: © 2011 Wired

Jerz, D. G. (2007) 'Somewhere Nearby is Colossal Cave: Examining Will Crowther's Original "Adventure" in Code and in Kentucky,' *Digital Humanities Quarterly*, 1 (2).
Link:
http://www.digitalhumanities.org/dhq/vol/001/2/000009/000009.html
Licence: This work is licensed under a Creative Commons Attribution-Noncommercial-No Derivative Works 3.0 License. This is an open-access article distributed under the terms of the Creative Commons Attribution-Noncommercial-No Derivative Works 3.0 License, which permits non-commercial use, distribution, and reproduction in any medium, provided the original author and source are credited.

Kitchin, R. (2012) 'The Programmable City, Environment and Planning B: Planning and Design,' 38 (6): 945-951.
Link:
http://www.envplan.com/epb/editorials/b3806com.pdf
Licence: © 2012 Pion. Made freely available on website.

Kovacs, G. (2010) 'Tracking the Trackers', TED.
Link to
Youtube: http://www.youtube.com/watch?v=f_f5wNw-2c0

Langner, R. (2011) 'Cracking Stuxnet, a 21st-century Cyber Weapon'.
Link to Youtube:
http://www.youtube.com/watch?feature=player_embedded&v=CS01Hmjv1pQ

Latour, B. (2012 forthcoming) 'The Whole is Always Smaller Than Its Parts - A Digital Test of Gabriel Tarde's Monads,' *British Journal of Sociology*.
Link: http://www.bruno-latour.fr/sites/default/files/123-WHOLE-PART-FINAL.pdf
Licence: © 2012 Bruno Latour. Pdf of pre-publication copy made available by author on website.

MacKenzie, A. (2006) 'The Problem of Computer Code: Leviathan or Common Power?' Institute for Cultural Research, Lancaster University. 10 August.
Link:
http://www.lancs.ac.uk/staff/mackenza/papers/code-leviathan.pdf
Licence: © 2006 Adrian MacKenzie. Pdf made available on authors' staff page.

MacKenzie, A. (2008) 'Wirelessness as Experience of Transition', *Fibreculture*, 13.
Link: http://thirteen.fibreculturejournal.org/fcj-085-wirelessness-as-experience-of-transition/
Licence: Creative Commons Attribution-NonCommercial-NoDerivs 2.5 Australia (CC BY-NC-ND 2.5). This is an open-access article distributed under the

Manovich, L. (2008) 'Software Takes Command'. *Software Studies*.
Link:
http://lab.softwarestudies.com/2008/11/softbook.html
Licence: Creative Commons Attribution-Noncommercial-No Derivative Works 3.0 Unported License. This is an open-access article distributed under the terms of the Creative Commons Attribution-Noncommercial-No Derivative Works 3.0 License, which permits non-commercial use, distribution, and reproduction in any medium, provided the original author and source are credited.

Marino, Mark C. (2006) 'Critical Code Studies', *Electronic Book Review*.
Link:
http://www.electronicbookreview.com/thread/electropoetics/codology
Licence: Creative Commons: Attribution-NonCommercial-ShareAlike 2.5 Generic (CC BY-NC-SA 2.5). This is an open-access article distributed under the terms of the Creative Commons Attribution-Noncommercial-ShareAlike 2.5 License, which permits non-commercial use, distribution, and reproduction in any medium, provided the original author and source are credited.

McCallum, L., and Smith, D. (2012) 'Show Us Your Screens'.

Link to Vimeo: http://vimeo.com/20241649

Matrosov, A., Rodionov, E., Harley, D. and Malcho, J. (n.d.) 'Stuxnet Under the Microscope'.
Link: http://go.eset.com/us/resources/white-papers/Stuxnet_Under_the_Microscope.pdf
Licence: © 2012 ESET North America. All rights reserved. Trademarks used herein are trademarks or registered trademarks of ESET spol. s r.o. or ESET North America. Freely Available White Papers

Najjar, M. (2012) Cover Image: 'The Sublime Brain [of Jonathon]'.
Link: http://www.michaelnajjar.com/
Licence: © 2011 Michael Najjar. Permission for use and reproduction of image as book cover of *Life in Code and Software* given by the artist. Original size: 150 x 150 cm.

Parikka, J. (2011) 'Media Ecologies and Imaginary Media: Transversal Expansions, Contractions, and Foldings,' *Fibreculture*, 19.
Link: http://seventeen.fibreculturejournal.org/fcj-116-media-ecologies-and-imaginary-media-transversal-expansions-contractions-and-foldings/
Licence: Creative Commons Attribution-NonCommercial-NoDerivs 2.5 Australia (CC BY-NC-ND 2.5). This is an open-access article distributed under the terms of the Creative Commons Attribution-Noncommercial-No Derivative Works 2.5 License, which permits non-commercial use, distribution, and reproduction in any medium, provided the original author and source are credited.

Parisi, L. and Portanova, S. (2012) 'Soft Thought (in Architecture and Choreography)', *Computational Culture*, 1.
Link: http://computationalculture.net/article/soft-thought
Licence: © 2012 Parisi and Portnova. *Computational Culture* is an online open-access peer-reviewed journal.

Ramsay, S. (2010) 'Algorithms are Thoughts, Chainsaws are Tools'.
Link to Vimeo: http://vimeo.com/9790850

Ramsay, S. (2011) 'On Building'. *Stephan Ramsay's Blog,* 1 November.
Link: http://lenz.unl.edu/papers/2011/01/11/on-building.html
Licence: © 2011 Stephan Ramsay

Slavin, K. (2010) 'How Algorithms Shape Our World', *TEDGlobal*.
Link:
http://www.ted.com/talks/kevin_slavin_how_algorithms_shape_our_world.html
Licence: © TED CONFERENCES, LLC

Turing, A.M. (1936) 'On Computable Numbers, with an Application to the Entscheidungs Problem', *Proceedings of the London Mathematical Society,* 2 (42): 230-65.
Link: http://turing.ecs.soton.ac.uk/browse.php/B/12
Licence: © 1937 London Mathematical Society. Made available via a link to the Turing Digital Archive.

Turing, A.M. (1950) 'Computing Machinery and Intelligence' *Mind*, 59: 433-460.
Link: http://turing.ecs.soton.ac.uk/browse.php/B/9

Licence: © 1950 Mind.

Weisstein, Eric W. 'Turing Machine', from *MathWorld - A Wolfram Web Resource.*
Link:
http://mathworld.wolfram.com/TuringMachine.html
Licence: © 1999-2012 Wolfram Research, Inc.

Wing, J. M. (2006) 'Computational Thinking',
Proceedings of the ACM, March, 49 (3): 33-35.
Link:
http://www.cs.cmu.edu/afs/cs/usr/wing/www/publications/Wing06.pdf
Licence: © 2006 Communication of the ACM. Made available on the author's university webpages.

Wing, J. M. (2009) 'Computational Thinking and Thinking About Computing'.
Link to Youtube:
http://www.youtube.com/watch?v=C2Pq4N-iE4I

why the lucky stiff (2009) 'Hackety Hack'.
Link to Vimeo: http://vimeo.com/5047563

Wolf, G. (2010) 'The Quantified Self', *TED.*
Link to Youtube:
http://www.youtube.com/watch?v=OrAo8oBBFIo