



Code/space and the challenge of software algorithms

Document Version

Accepted author manuscript

[Link to publication record in Manchester Research Explorer](#)

Citation for published version (APA):

Dodge, M. (2016). Code/space and the challenge of software algorithms. In B. Warf (Ed.), *Handbook on Geographies of Technologies* (pp. 1-30). Edward Elgar Publishing Ltd.

Published in:

Handbook on Geographies of Technologies

Citing this paper

Please note that where the full-text provided on Manchester Research Explorer is the Author Accepted Manuscript or Proof version this may differ from the final Published version. If citing, it is advised that you check and use the publisher's definitive version.

General rights

Copyright and moral rights for the publications made accessible in the Research Explorer are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Takedown policy

If you believe that this document breaches copyright please refer to the University of Manchester's Takedown Procedures [<http://man.ac.uk/04Y6Bo>] or contact uml.scholarlycommunications@manchester.ac.uk providing relevant details, so we can investigate your claim.



Draft chapter for Handbook on Geographies of Technologies, edited by Barney Warf. May 2016

Code/space and the challenge of software algorithms¹

Martin Dodge

Department of Geography, University of Manchester

“Software has replaced a diverse array of physical, mechanical, and electronic technologies used before 21st century to create, store, distribute and interact with cultural artifacts. It has become our interface to the world, to others, to our memory and our imagination - a universal language through which the world speaks, and a universal engine on which the world runs.”
(Manovich, 2013)

How many waking hours in a day are you focused on your mobile phone, a laptop screen, the smart television on the wall, a connected watch on your wrist or the GPS display and digital dashboard in your car? Lives are lived through computer screens, which are really windows into a space of code. The spaces are brought into being through software algorithms working on data in ways and at speeds that are hard, if not impossible, for people to comprehend. It's not possible to really see code as code and even if we did few would have the capacity to meaningfully decipher it. Yet code is active and important social actor, it is indispensable and unavoidable in the conduct of daily life. Code change how space is produced and poses a significant challenge to unpack and explain.

Myriad systems of software are now essential to the functioning of many societies. Software makes a difference to how social, spatial and economic life takes place. It is a vital element in the operation and governance of transport and utilities infrastructures, the planning and maintenance of city spaces, consumption practices, work processes in virtually all sectors of the economy, as well as personal life in domestic settings. To date, most analysis by geographers and allied social scientists has been focused on the technologies that software

¹ Some of the material discussed in this chapter is based on collaborative research with Rob Kitchin and published in our 2011 book *Code/Space* (MIT Press) and other papers.

enables, rather than the underlying code (data inputs, information stores and decision-making algorithms) that does work in the world and is capable of bringing particular socio-spatial formations into being.

An indicator of the contemporary significance of code is the huge economic value accruing to most successful companies that develop software and provide digital services. Software businesses have grown rapidly and come to rival and even eclipse established corporate giants in petroleum production, retail and automobile manufacture in terms of stock market capitalization. Corporations like Microsoft, Alphabet (parent company of Google), Facebook, SAP, Oracle and Apple are worth tens of billions and often highly profitable. For two decades Microsoft was the largest software firm, by a significant margin, and it still had revenues in 2015 of \$93.6 billion and had a market value of \$416b (March 2016). In terms of economic scale (and media profile and cultural cache) Microsoft has been usurped by the likes of Google in recent years. The new parent company Alphabet has only 61,000 employees but its algorithms and vast databases are depended on by hundreds of millions of people every day to solve all manner of tasks and consequently investors value the business massively (capitalization was \$516b in March 2016).

Beyond the influence of individual market leading tech corporations, software is elemental new rounds of capitalism seen most overtly in the dot-com boom in the late 1990s, the rise of social network more recently and the current rapid growth in the so-called 'sharing economy', which is dramatically reshaping established markets. For example in fields of urban mobility and short-term rental space the success of new economic models from companies like Uber and Airbnb rest solely on their innovative code with dynamic databases of available resources and clever matching algorithms often exploiting on geolocation information from smartphone apps, along with seamless online payment systems and means of creating trust in virtualized exchange through reviews and recommendations.

There is a need to be alert to the organization cultures, political economy and the geographical locations to the development of the software underpinning social networking, smartphones and the sharing economy. The code is conceived and written by people, usually employed by corporations, and working in particular places. Consequently software is not neutral agent; it embodies the values and biases of those who write it. Work by

economic geographers and regional scientists shows how much the creation, maintenance, and marketing of software products is heavily concentrated in relatively few places and conducted by highly specialized workforce, despite the rhetoric that often surrounds 'tech companies (e.g., Andreosso-O'Callaghan et al, 2015; Arora and Gambardella, 2006; Cook 2013). Distinct but closely interconnected to power of software there has been massive growth in collection of near continuous streams of personal data on millions of people. Whenever a person interacts with software system - every login, swipe of a card, social network post, online payment, opening of an app, clicking on a hyperlink - is logged and often tied to identifiers and geolocation details. According to the influential security consultant Bruce Schneier (2016, no pagination), "[d]ata brokers save everything about us they can get their hands on. This data is saved and analyzed, bought and sold, and used for marketing and other persuasive purposes. And because the cost of saving all this data is so cheap, there's no reason not to save as much as possible, and save it all forever." As such what has been termed a 'data shadow' follows people and informs how they are treated in future interactions with businesses, government agencies and other institutions that have access to various profile and risk scores. There is scope more research by human geography and other, examining how the labor of software writing and collection of personal data is both placed and scaled, shaped by cultural values and other contextual factors.

It is clear that software enabled technologies, digital services and social networks, and new kinds of ecommerce predicated on vast silos personal information have brought many benefits, although these are not shared equally or available to all. However as lives become more dependent on correct operation of software and automated evaluation records in data shadows there are growing risks and scope for new kinds of malfeasance and criminality. One way that these risks arising from increasing dependency software have been highlighted through impacts of thefts of personal information from network databases, often compromising the privacy of millions of people and putting them at danger of identity and financial frauds. Some most high profile recent cases include the hacking of systems of online dating website Ashley Madison and the public release of confidential details on the members (cf. Hern, 2015), repeated break-ins to customer database of the telecommunications company TalkTalk (cf. BBC, 2015) and the politically damaging theft of

many millions of from the US Office of Personnel Management with consequences for personal safety and national security (cf. Davis, 2015). Beyond thefts of data, many analysts point to escalating risks from deliberate sabotage of software, with attacks conducted virtually but causing real-world disruption to crucial systems and infrastructure. A potent recent example is the sophisticated cyber-attack on the software controlling the electricity grid that blacked out large parts of Ukraine in 23 December 2015 (Zetter, 2016). The motives of the attackers in this case are unclear but given the geopolitics of this region suspicion has focused on role of the Russian state in sponsoring or directing the sabotage. It seems that cyberwarfare fought through computer networks using code, but with impacts on cities and essential services, that was once the realm of science fiction and fear mongering speculation is fast becoming a reality.

Figure 1. Control rooms with banks of screens to monitor myriad processes and specialized server computers in data centers. Such spaces are essential to contemporary society but often hidden away from sight, with restricted access they are designed specifically for work of software and data computation, their functional value *depends* on code. They are code/space.

(Source. Top image by Maximilien Brice, 2008, courtesy of CERN, <https://cds.cern.ch/record/1125846>. Bottom image by Leonardo Rizzi, 2010, www.flickr.com/photos/stars6/4381851322/)

Discourse supporting software

It is important to realize that there are distinct sets of discourses that are at work promoting ever deeper algorithmic automation of society and counter-acting criticism of code and risks it brings. Computer technology more generally but software specifically is subtly, and not so subtly, advocated as the *only* possible way to solve contemporary social problems by many actors (e.g. security from terrorism can only be tackled by more software surveillance and monitoring of online activity) Software is promoted endlessly as *essential* means to keep economies competitive in the 'global race'. It is important to interrogate the discourses that are constructed by companies with direct vested interests, governments and other institutions (including universities) to create the discursive regime that supports and

normalizes the development and roll-out of software dependent infrastructures and processes.

Above all the discursive regime of code seeks to present a commonsensical façade – it simply ‘makes-sense’ to invest in software technologies. Using lobbying, media campaigns, direct advertising, inducing the early-adopters, trendsetters and celebrity endorsements, and employee training and educating children, the argument for entrusting more and more of everyday life and its governance by software systems is made with reference to incontrovertible issues of efficiency, making money, time-savings, security and safety, personal empowerment. (Such discursive regimes are produced by vested interests with respect to the notions of ‘smart homes’ and ‘monitoring of bodily performance’, as discussed below.)

The power of discourse, as Michel Foucault and others have point out, lies in persuading people to its logic—to believe and act in relation to this logic – and without thinking there might be alternative ways precede. And with social conventions and cultural norms of state capitalism it can be hard to resist such logic that technology can make society more secure and productive. These discourses are often promoted by key actors in government in tandem with business elites and consultants, driven by the interests of corporate capitalism and, increasingly, the agenda of neoliberalism focused on exploiting software as far as possible to cut-costs, outsource, de-skill and reduce staffing, reconfigure public services for profit within a target-driven culture. People who cannot or choose not to participate are easily marginalized (e.g. requirements to register for welfare services through software systems, or being excluded from cheaper prices from online only services channels)

Importantly the discursive regime for more deployment of software does not operate solely from the top downward, but through diffused microcircuits of power, the outcome of processes of local regulation, self-regulation, and small notes of resistance. As such, people are not simply passive subjects drawn along in unproblematic ways by desire for evermore software. Notes of contestation are made by civil rights activists and privacy groups, skeptical journalists and critical academic voices, more broadly there is a kind of passive reluctance and apathy in regards to exaggerated claims for software (e.g. around ease of use or productivity gains). Although it has been argued, given the increasing power and role of

software, that resistance to digital technologies has been remarkably mute; as Thrift and French (2002, 313) note, “[e]ven though software has infused into the very fabric of everyday life—just like the automobile— it brings no such level of questioning in its wake.”

The nature of code and need for ‘software studies’

Software consists of lines of code—specific decision-making instructions and mathematical algorithms that, when combined and supplied with appropriate input data, produce routines and programs capable of complex digital functions and crucially performing appropriate [‘correct’] actions. Put simply, the action of software can instruct computer hardware (the physical, digital circuitry) in an unambiguous way about what to do (which in turn can engender action in other machinery, such as switching on electrical power). Although code in general is hidden, invisible inside the machine, it can produce visible and material effects in the world.

From humble beginnings in the 1940s, software can now be hugely complex and in many, varied forms, from abstract machine code and assembly language (‘low level’) to libraries and APIs and up to more formal programming languages, applications, user-created macros, and simple scripts (‘high level’). (Figure 2) Often these forms are nested together or arranged into hierarchically connected libraries of code to produce effective entities. These might be hard-coded applications embedded on chips, specialized applications (banking software, traffic management systems), generic user applications (word processors, Web browsers, video games), and large operating systems (Windows, Mac OS, Linux), that run on a variety of hardware platforms (smartphone, dedicated units, PCs, server).

Figure 2. High-level representations of code and the juxtaposition have been chosen to illustrate the scale of change. On the left is a handwritten notion of what is claimed to be one of first computer programmes, written by Tom Kilburn 1948 as part of the pioneering computer research at the University of Manchester. On the right is the ‘Linux Kernel Map’, created by Constantine Shulyupin, displaying the hierarchy of software components at the core of this operating system and how to work together, (

(Source: Left image courtesy of G. Tootill. Right image courtesy of Constantine Shulyupin, www.MakeLinux.net/kernel_map)

The phenomenal growth in software development and deployment stems from the fact that digital code (unlike analogue forms of instructions) has executable properties, that is, how it codifies the world into rules, routines, algorithms, and databases and then uses these to do work in the world. This means it can run by itself, and although software is not sentient or self-conscious, it does exhibit some of the characteristics of being alive. Thrift and French (2002, 310) describe self-operative nature of software as being “somewhere between the artificial and a new kind of natural, the dead and a new kind of living” and having “presence as ‘local intelligence.’” This property of being ‘alive’, (executable and self-active) is significant because it means code can make things do work in the world in an autonomous fashion—that is, it can receive inputs and process data, evaluate diverse situations, reach complex decisions, and, most significantly, act without human oversight or authorization. As an illustration of this, at any moment when you are interacting with a laptop or smartphone there are likely hundreds of other pieces codes and larger programmes performing their own work beyond human awareness (Figure 3). Moreover code had pervaded into non-computer like objects and everyday environments in often subtle and opaque ways, so software is forming a ‘technological unconscious’ that is noticed only when it performs incorrectly or fails (Graham and Thrift, 2007). Another important, related development is that of machine-to-machine, communications with smart devices and tagged objects connected to through the so-called ‘internet of things’ which is creating a parallel digital ecology that exists without human authorization or even awareness to what is happening. A consequence, of evolving power of software and spatial diffusion is things people interact appear to be ‘automagical’ in nature in that they work in ways that are not clear and visible to most people, and it produces complex outcomes that are not easily accounted for by their everyday experience of the analogue and non-digital.

Figure 3. A display of the numerous separate piece of software running on a typical laptop (controlled by Windows7 operating system) and various installed applications and utilities. Only a few are directly initiated by the person using the laptop and much of what is running is not meant to interact with people at all. (Source. Author image)

Executable code with ‘automagical’ algorithms for reaching automated decisions that must be correct and unchallengeable [within its operating domain and input parameters] has

clear social and political implications. And yet the internal workings of software has been poorly theorized and empirically little studied from a social sciences perspective beyond a handful of formative texts by cultural and new media theorists (including, Manovich (2013), Fuller (2008), Galloway (2004), and Mackenzie (2010)). Instead, software has been understood from a technical, instrumental perspective that treats it as largely an immaterial, stable, neutral product, rather than as a complex, multifaceted, mutable set of relations created through diverse sets of discursive, economic and material practices.

However the last few years efforts have been made to develop 'software studies' as a field of scholarly enquiry that seeks to create an expanded understanding of code that extends significantly beyond the technical and examine culture of digital computation. It is largely culturally-informed epistemologies and theoretical critiques to ask how the social world itself is captured within code in terms of algorithmic potential and formal data descriptions. Software studies focuses attention squarely on the nature of code and not simply its effects, and it conceptualizes software as both a product of the world and a producer of the world in a way that recognizes that its production and work is not deterministic and universal in form. According to Lev Manovich (2013, 10), an influential theorist in the field, "I think that Software Studies has to investigate both the role of software in forming contemporary culture, and cultural, social, and economic forces that are shaping development of software itself." From a geographical perspective, it also means recognizing that potential work of software is contingent on spatial context and is the product of people in time and particular places, furthermore its use can transform these times and places, and as such the same code may work in different ways in different grounded situations.

Matthew Fuller (2008) argues software studies proposes that code can be seen as an object of study and an area of practice for kinds of thinking and areas of work that have not historically authored or owned software, or indeed often had much to say about it. In this regard there is much that needs to be said by geographers who have traditionally not had much to say about the spatiality of software. One way to begin this is to analyze the way in which code can, quite literally, bring unique kinds of spaces into being. In some circumstances software is enrolled so completely into sociospatial relations that they are

dependent on the effective operation of code; these are what Dodge and Kitchin (2005) have called code/spaces.

The notion of code/spaces

Here we shall consider the conceptual model of code developed by the author working with Rob Kitchin. As a kicking off point Kitchin and Dodge (2011) made the case that to understand this large-scale structural change flowing out of the technicity of software, one needs to develop rich historico-geographical accounts of the contexts in which code is embedded into workplaces and labor practices over the time and the new kinds of future trajectories this enables. Otherwise, code is simply seen as an abstract, exogenous factor rather than a socially embedded variable.

To develop their account of code Dodge and Kitchin (2005) advanced the ideas that software can be embedded in everyday life at series of scales / levels of activity. They proceeded firstly by defining the range of forms of software into a four-level hierarchy: (i) individual coded objects, which can be groups together or linked to form (ii) coded infrastructures, which in turn are controlled by and also carry (iii) coded processes. Coded objects, infrastructures and processes are in turn, can combine together to form larger (iv) coded assemblages. This hierarchy enables the software, through its varying degrees of technicity (power and productive capacity for work) to transduce space, that is it brings new spatial formations into existence to solve a problem or perform a task. Furthermore these spatial transductions through the enrolment of software, we theorized at three distinct levels of intensity: (i) code/space, (ii) coded space, (iii) background coded space.

In the most intensive level of transduction the technicity of software is so significant that the space brought into being *depends* on the operation of the code. There is a dyadic relationship between the existence of the space and the execution of the code – hence the co-joint term ‘code/space’ – and if the software fails to operate then the space is not produced. Code/space occurs when software and the spatiality of everyday life become

mutually constituted, that is, produced through one another. Here, spatiality is the product of code, and the code exists primarily in order to produce a particular spatiality. People regularly coproduce code/spaces, even if they are not always aware they are doing so, and as we consider in the next two empirical sections of this chapter, they are increasingly common in a range of everyday contexts and domestic situations (cf. Dodge and Kitchin, 2009). Any space that is *dependent* on software-driven technologies to function as intended constitutes a code/space: workplaces dependent on office applications such as spreadsheets, shared calendars, information systems, networked printers, e-mail, and intranets; aspects of the urban environment reliant on building and infrastructural management systems; many forms of transport, including nearly all aspects of air travel and substantial portions of automobility and rail transport (Figure 4); and majority of the components of the telecommunications, media, finance, and entertainment industries. Such is the reliance by governments and businesses on a raft of office applications and larger software systems that it is now unthinkable to backtrack to a pre-digital paper-based processes: the nature of tasks has changed, staff levels have been reduced and deskilled in many cases, and operational networks and transactions have become much more complex and interdependent.

Figure 4. Many of the places that people live in; the offices, shops, and factories they work in; and the vehicles they travel in are code/spaces, at least part of the time. A particularly high-profile would be the jetliner which is code/space with flight *dependent* on multiple software systems and distinct coded objects. The display screens in the cockpit are small portals into the larger realm of code that monitors and controls much of the aircraft's performance, and software is primary determinant on how the pilots work. (Source. Guillaume Grandin , www.guillaumegrandin.eu)

In the second level of intensity, what we termed 'coded space', the transduction is mediated by code but the relation is not dyadic so if the software were to fail to operate for whatever reason the space would still be produced as intended to solve a problem or perform a task. However, the nature of the spatial transduction without software is potentially a less efficient solution to the problem or a more costly way to perform a task (e.g., failure of computer system forces workers to use a 'manual' backup procedure that is much more labor intensive, slower and more costly, or perhaps occurs with reduced safety). Here, the role of software is often one of augmentation, facilitation, monitoring,

and so on rather than control and regulation.

The lowest intensity might be thought of as a transduction-in-waiting, what Dodge and Kitchin (2005) termed 'background coded space', a situation when software is present in the environment and has the potential to mediate a solution if activated. Code could make a difference but is inert unless consciously evoked. Much of people's ordinary living in Western cities occurs in 'background coded space' where people are surrounded by coded objects, coded infrastructures and coded processes that can be called upon in myriad of ways to solve a problem or perform a task. For many people much of their deliberative daily activities precede in coded space —that is, places animated by software and where their practices are routinely augmented by algorithms and software without human intervention. It can plausibly be argued that an increasing amount of time, for a larger range of individuals, is spent within bubbles of code/space where the spatial transduction depend wholly on software to achieve the tasks and solve problems they face.

Delineating code: (i) domestic automation and everyday living

The significance of code might be more obviously apparent in 'high-tech' public spaces, at airports, in contemporary office with a computer display on every desk or the robotic production lines in highly automated manufacturing plants, but software is being enrolled evermore deeply to transform the spatiality of people's homes and daily self-reproduction of their bodies. The domestic realm has become coded space for most people living in countries in the Global North and code/space for some.

Throughout the modern period the home has been a site of technologies to augment daily activities and, supposedly, to reduce the burden of domestic labor through automation. Like earlier rounds of electromechanical appliances and electronic gadgets that were initially a novelty and luxury goods, but became taken-for-granted and sunk into background of most domestic spaces, so software-enabled technologies have been transforming the performance of home life over last twenty years or so, helping people solve everyday tasks (such as cleaning, cooking, recreation, shopping, saving and sharing familial memories,

maintaining personal hygiene and well-being). Most homes today are populated with lots of distinct coded objects, a good number of which are becoming connected to together (by wifi and other radio communication) and able to interact with each other and outside companies and institutions through coded processes. Most particularly for those with money and the inclination then much of the 'smart home' fantasies from the early decades are becoming a reality, although the degree to which the latest digital gadgets and software-enabled appliances are really improving people's lives and making their home happier place is debatable. The promise of more leisure time, particularly for women who conventionally carry most of the burden of maintaining the home, through purchasing new appliances is a longstanding myth (cf. Cowen 1983); and if anything adding software to domestic spaces simply adds in a new layers of complexity and stresses! (Think of new burdens about software security, remembering passcodes, changing passwords, ensuring compatibility and needs to update this and that to keep it working; such work was never needed with an analogue television set!)

While we need to be critical of upbeat, almost utopian, predictions of better living in software-enabled 'smart homes' that are promulgated powerful by vested commercial interests, there is no doubt that code has come into the homes in developed countries in significant ways. An overt example of the potential of code object to change domestic space into coded space is evident in deployment small home robots designed to perform routine tasks in largely autonomous way using software algorithms and awareness of aspects of its environment from sensor inputs. (Figure 5) While such robots remain rather a gimmick, more prosaic and potent impact of code is in enhancing home entertainment technologies and the depth of consumption of digital media over last decade; for many people key aspects of their recreation at home is now dependent on software and coded objects. (Although as with any technological evolution there is never a complete substitution of one form for another, and it is likely there are a good number of homes happily using analogue VHS videos and or reveling in the cult status of vinyl records!) Today, the domestic space of affluent households are literally overflowing with all kinds of computer devices for delivery entertainment content and keeping people 'occupied' (and consuming): videogame consoles, interactive toys, digital radios, tablets, laptops, webcams, camcorders, media

players, ebook readers, and so on. Screens in almost every room, including the bathroom and many people take their smartphone or tablet into the loo!

The television set, the central technological component of domestic recreation for the majority of homes since the 1960s, has become a 'smart tv' with powerful software, added features and complex screen interfaces. In the UK the last broadcast analogue television ended in 2012, making this media only available digitally and dependent on coded objects to process the signal, while in last few years more people have stop following fixed television schedules and become consumers of on-demand streaming media, which depends wholly on software to deliver the right show to the viewer. As such to watch television is to bring into being code/space. There is claim for more choice and interactivity but new, so-called 'smart tv' sets with two-way connection to internet also require 'looking after' in ways that never needed with 'dumb' cathode ray tube set plugged into analogue aerial socket. Functionally many of latest television sets log usage patterns sets, and some have environment sensors to make them aware of their surroundings and how they are being used, couple with the ability to send this data out of home as well as receiving media. A particular risk was flagged those smart tv sets with voice activated controls because they listening to all in room conversation to offer enhanced usability but with concomitant concerns about privacy of the domestic realm (Schneier, 2015). Television sets use code to watch and record the watching household and there is an analogy to be drawn here to the 'telescreens' imagined by Orwell in his dystopian novel *1984*.

Overt coded objects like large smart televisions hanging in pride of place on the living room wall, white appliance with digital controls in the kitchen and clever little robots for cleaning are easily visible instance of software in home but perhaps more significant will be the enrolment of software to provide a management system of the whole domestic space and environment, potential making the whole assemblage into a code/space. More sophisticated building management systems (which have typically been associated with high-end office complexes and the like) are now being heavily promoted to the domestic market with the ability to coordinate all manner of mundane but vital tasks, heating and air conditioning, access and security monitoring, fire/flood warning through sensors, and smart meters able to adaptively tune utility usage. Such home 'hubs' promise more control, for example residents can have remote access to services at home from smartphone app (e.g.

tweaking the heating before arriving home), and also ease-of-use as the software will take care of things for you through self-learning of household patterns and auto-scheduling services. Promotional rhetoric rests on powerful discourses of efficiency, comfort and risk-reduction. For example, one of the current market leading 'smart thermostat' products NEST asserts that its controller system "uses state of the art machine learning algorithms to determine the thermal properties of each home individually. And once the thermostat understands how the house warms up and cools down, it allows it to keep users much more comfortable while maximizing their energy savings." (NEST 2015, p.3) The understanding of physical environment of individual homes and the daily behavior patterns of the household through software in such management systems could well lead to more efficient energy usage and some saving of money but it comes at the cost with an intensive 'spy' in the home, compromising the accepted private sanctity of domestic space by sending out detailed consumption of water and power to utilities companies. This is coupled with a significant risks that such home management hubs, accessible through the internet, could be hacked by those with malicious intent (Hernandez et al, 2014).

Figure 5. The possibility of software to make life easier, as seen in two magazine advertisements promoting domestic robots to potential purchasers. The claims to be able automate specific tasks that are typically seen as onerous and thereby offering the owners prospects of more 'free time' is long standing technological discourse and still a powerful one. (Source. Author images)

While there is scope to be critical of these home management systems and claims made from the vendors, there is a strong case to be made that safer living through software in terms of home telecare systems will become ever more significant in next decade (although with attendant risks and great loss of privacy.) A key challenge facing many affluent developed countries is coping with much larger aged populations, including many physically frail and cognitively forgetful people who desire to live independently in their own homes for as long possible. Telecare systems are promoted as a solution with software able to continuously, automatically and remotely monitoring daily behavior looking for changes in routines and also be ready to respond to real time emergencies (pressing a wearable panic alarm). For example detectors on fridge door, kettle and oven operation provide data

streams for software algorithms to process and give indication that the householder might be failing in their ability to prepare hot drinks and daily meals. (Figure 6) As more sophisticated telecare monitoring systems become available so (affluent) pensioners and the disabled in particular can live in evermore coded cocoon. They may feel safer and potentially are kept safer but it is less clear how well code can really care for people beyond monitoring their physical performance and material circumstances.

Figure 6. (left) An info-graphic from a marketing brochure of a leading domestic telecare system in the UK indicating the range of possible routine activities that are monitored by sensors linked to a local control box and onwards to software systems in distance control center. (right) Image of the kind of temporal pattern analysis that can be presented by the telecare system to monitoring staff as tool to spot behavioral change that may indicate a problem with the person. (Source. Author images)

Software has crossed the threshold and infusing into the large majority of homes in significant but still partial way, and while its presence will undoubtedly grow it will remain haphazard and incomplete. Some of it also strongly tied to income levels and social status in buying into the latest rounds of software technologies and 'smart' gadgets. Much is coded space in which the activity could continue even if the software monitoring failed and the log was not updated, although watching your favourite tv series on a tablet through a streaming service is code/space.

Delineating code: (2) performing software surveillance of the self

Of course code is not just monitoring the health and wellbeing of frail elderly people via expensively installed telecare system, there is now a whole plethora of wearable gadgets and software apps for those interested in self-surveillance of their bodily performance. (Figure 7) Wristband trackers are marketed for serious sports types, for those keen to improve their fitness or just daily prurient fascination with how far they have walked in last 24 hours, how many calories were nominally burnt and their sleep pattern over the previous

week. This new breed of coded objects key into all kinds of consumerist trends around health and body and technological design meant appeal to the affluent and gadget obsessed, with some promoted by existing high-profile brands (e.g. Nike with its Sportband) and others are cool companies with 'must-have' cache (like the Jawbone UP or Fitbit Flex). Evaluation of users shows the act of tracking becomes important in and of itself, and the build-up of log of performance become information to play with, contemplate in idle moments and perhaps worry over ('why has my fitness level dropped this month?'). (cf. Rooksby et al, 2014; Williamson, 2015). The tracker hardware communicates seamless to software on the owners smartphone and the attendant app does not just present the data, algorithmic capacity means it act as an virtual trainer able to encourage greater efforts and actively making suggestions on how to hone bodily performance in the future. Another aspects, made possible by software, is the sharing bodily performance with family, friends or peer groups online; this can engender pride, sense of self-worth or just be about ego and competition.

Wristband has been a rapidly developing market segment in recent years as more people see the appeal of so-called 'self-quantification', but in many ways it merely a parallel to much more significant and more medically determined monitoring of the body through software (e.g. coded objects and coded processes enrolled in management of chronic morbidity in the home settings such as heart conditions with blood pressure testing and diabetes with blood sugar testing pens.) (cf. Lupton, 2013) Of course intimate body monitoring, for example using scales to watch weight changes, is a long standing practice for many but is now enhanced with code as digital data that can easily be stored and graphed for trends on a smartphone app; for many this kind of algorithmic analysis of their body will be helpful but for others more likely anxiety inducing, with data displays keying into all kinds of latent worries around diet, (un)healthy eating and the specter of obesity. Other examples include personal air pollution sensors aimed at the growing number of asthma sufferers, and for those seeking to get pregnant the use software to better monitor menstruation and ovulation cycles (perhaps a positive result regarded as coded conception!). More generally the algorithms in smartphone apps have enabled rapid development of the home medical market and allowed for an intensive degree of self-monitoring and wider collection of health status data. In 2012, 69% of smartphone owners in America reported tracking at least one

health indicator such as weight, diet or exercise (Fox and Duggan, 2013). The resultant data can be exploited to trigger highly personalized interventions and can be stored in large databases with the potential to improve healthcare research and plan better public health strategies. There is also huge commercial interest in such individual and detailed data.

Figure 7. (left) Product manufacturers seek new rounds of consumption by linking into the self-surveillance of the body, with electric toothbrush morphing into a code object with promise better oral hygiene with software monitoring and regulating the person's performance. (right) An example of the promotional rhetoric for Fitbit, currently one of leading wearable personal performance trackers. (Source. Author images)

Epilogue: further challenges in comprehending code/space

Further analysis by human geographers and other social scientists to understand more fully the nature of software and where coded space occurs and how of code/spaces come into being will require investigations in specific places and developing means to interrogate how algorithms are working in everyday contexts. This will be a challenge for several reasons, firstly because the software sector is so dynamic and exhibits such fierce competition between corporations (for example in smartphone apps), this means there is a fast turnover in applied code and updates in algorithms. Considered scholarship struggles when the target of analysis never stands still. Moreover much of this executable code is proprietary and owned by corporations that, understandably, want to keep the operations of their key algorithms as trade secrets for fear of losing competitive advantages or being exposed for having questionable products (for example code that makes judgments that unwittingly discriminate against some users, or are deliberately biased in generating results for commercial gain or erroneous in particular circumstances, or simply with unfixed security flaws; cf. Kirchner 2015; Ziewitz 2016; Zook and Graham, 2007). The temptation will be to analyze publicly available forms of code (such as online media and open source software development) rather than investigating the harder to access code that matters most to daily life and the ongoing production of the society (e.g. control systems of the utility companies, the scheduling software of transport systems, the tasking of security personnel, the calculation of insurance rates and mortgage risks, etc.).

Beside speed of change and the issue of corporate secrecy, the core code at the center of major software products are archetypical 'Black Box' operations with data going in and results coming out but no detail on the interior workings. And even if researchers were able to break open the 'black boxes' of commercial software it is not clear whether they could really interpret what they find inside. While there might be valid political reasons to seek such algorithmic transparency, there are likely to be practical issues of how to read and understand what is there. Without significant experience in mathematics, coding architectures and arcane syntax, even for very diligent researchers the algorithms will remain unfathomable. (As a simple illustration of the scale of the interpretative challenge – just look at the scripting behind a typical webpage and see if you can work out what is it doing and how this relates to what you see in the browser; Figure 8). Assuming social science researchers could develop the necessary vocabulary and grammar to begin to decode the obtuse workings of algorithms at the heart of software systems, like a smartphone, there is no guarantee that this will produce meaningful knowledge in a social science sense. Will identifying individual decision points and operations in an algorithm provide recognizable attributes with social context or relatable properties to use as handholds for critical geographical analysis? As Mackenzie showed in his analysis of digital signals processing performed in mobile telephones (2009, 1295) “the algorithmic processes ... offer a strong challenge for research. ... in their somewhat stunning complexity, they seem to bear only a tangential relation to the powerful dynamics of belonging, participation, separation, and exclusion typical of contemporary network cultures.”

Besides trying to examine source code directly, Kitchin (2014, 17-22) suggests some other ways of researching algorithms, and their wider significance, including researcher reflexively producing their own code, reverse engineering, interviewing designers and conducting ethnographies of coding teams, and examining how algorithms do work in specific places. Even if such research becomes possible, a further problem is how to communicate the workings of an algorithm to a wider audience without resorting to mathematics or repeating the logical operations in slightly different annotated form. An effort in this direction by Cormen (2013), is worthwhile but demonstrates how hard it is to do in practice to represent

and explain algorithms to the layperson. There may also be scope for applying spatial visualization to map out the flows of data logic of decisions made in a given algorithm.

Figure 8. The view of the source code for a typical webpage. Shown here is small part of the complex scripting and opaque relation to how the actual information content appears to the user. (Source. Author image)

Ultimately, the challenge to understanding software is hard going for human geographers because overwhelming majority cannot code! While academic geographers are very much software-embedded workers, like many other occupations and professions, they generally do not need to write code to tackle their daily scholarly tasks. This inability to programme software (held generally across humanities and social sciences) is a problem as there is a real danger of becoming thoroughly alienated from a key source of creative power in working practices going forward. As van Kranenburg (2008, 23) put it, in an analogous context:

“If as a citizen you can no longer fix your own car -which is a quite recent phenomenon -because it is software driven, you have lost more than your ability to fix your own car, you have lost the very belief in a situation in which there are no professional garages, no just in time logistics, no independent mechanics, no small initiative. ... [Citizens] become helpless very soon, as they have no clue how to operate what is running in the background, let alone fix things if they go wrong. As such, ‘ambient intelligence’ presumes a totalising, anti-democratic logic.”

So while geographers are making some progress in describing the consequences of code for everyday activities but before proceeding much further they will need to take some courses in computer science to really begin expose algorithmic pinch-points and explain in precise ways how a piece of software, working in a particular place, can bring code/space into being.

References

- Andreosso-O'Callaghan B, Lenihan H, Reidy P, 2015, "The development and growth of the software industry in Ireland: An institutionalized relationship approach", *European Planning Studies*, 23(5): 922-43.
- Arora A, Gambardella A, 2006 *From Underdogs to Tigers: The Rise and Growth of the Software Industry in Brazil, China, India, Ireland, and Israel* (OUP).
- BBC News, 2015, "TalkTalk cyber-attack: Website hit by 'significant' breach", *BBC News Online*, 23 October, www.bbc.co.uk/news/uk-34611857
- Cormen T H, 2013 *Algorithms Unlocked* (MIT Press).
- Cowen R S, 1983 *More Work for Mother: The Ironies of Household Technology from the Open Hearth to the Microwave* (Basic Books).
- Cooke P, Glen Searle G, O'Connor K, 2013 *The Economic Geography of the IT industry in the Asia Pacific Region* (Routledge).
- Davis J H, 2015 "Hacking of Government computers exposed 21.5 million people", *New York Times*, 9 July, www.nytimes.com/2015/07/10/us/office-of-personnel-management-hackers-got-data-of-millions.html.
- Dodge M, Kitchin R, 2005, "Code and the transduction of space", *Annals of the Association of American Geographers*, 95: 162-80.
- Dodge M, Kitchin R, 2009, "Software, objects, and home space", *Environment and Planning A*, 41: 1344-65.
- Fox, S, Duggan M, 2013, 'Tracking for health', *Pew Research Center*, 28 January, www.pewinternet.org/2013/01/28/tracking-for-health/.
- Fuller M, 2008 *Software Studies: A lexicon* (MIT Press).
- Galloway A R, 2004 *Protocol: How Control Exists After Decentralization* (MIT Press).
- Graham S, Marvin S, 2001 *Splintering Urbanism: Networked Infrastructures, Technological Mobilities and the Urban Condition* (Routledge).
- Graham S, Thrift N, 2007, "Out of order understanding repair and maintenance", *Theory, Culture & Society*, 24(3): 1-25.
- Hern A, 2015, "Infidelity site Ashley Madison hacked as attackers demand total shutdown", *Guardian*, 20 July, www.theguardian.com/technology/2015/jul/20/ashley-madison-hacked-cheating-site-total-shutdown.
- Hernandez G, Arias O, Buentello D, Jin Y, 2014, "Smart Nest Thermostat: A smart spy in your home", Blackhat Conference, 2-7 August, <http://www.blackhat.com/docs/us-14/materials/us-14-Jin-Smart-Nest-Thermostat-A-Smart-Spy-In-Your-Home-WP.pdf>

Kirchner L, 2015, "What we know about the computer formulas making decisions in your life", *ProPublica*, 30 October, <https://www.propublica.org/article/what-we-know-about-the-computer-formulas-making-decisions-in-your-life>.

Kitchin R, 2014, "Thinking critically about and researching algorithms", *The Programmable City Working Paper*, Maynooth University, Ireland, <http://papers.ssrn.com/abstract/2515786>.

Kitchin R, Dodge M, 2011 *Code/Space: Software and Everyday Life* (MIT Press).

Lupton D, 2013, "Quantifying the body: monitoring and measuring health in the age of mHealth technologies", *Critical Public Health*, 23(4):393-403.

Mackenzie A, 2009, "Intensive movement in wireless digital signal processing: from calculation to envelopment", *Environment and Planning A*, 41(6): 1294-1308.

Mackenzie A, 2010 *Wirelessness: Radical Empiricism in Network Cultures* (MIT Press).

Manovich L, 2013 *Software Takes Command* (Bloomsbury Academic).

NEST, 2015, Thermal Model and HVAC Control White Paper, <https://nest.com/downloads/press/documents/thermal-model-hvac-white-paper.pdf>

Rooksby J, Rost M, Morrison A, Chalmers M, 2014, "Personal tracking as lived informatics", *CHI 2014 Conference*, 26 April - 1 May, Toronto. <http://dx.doi.org/10.1145/2556288.2557039>.

Schneier B, 2015, "Your TV may be watching you", *CNN*, 12 February, <http://edition.cnn.com/2015/02/11/opinion/schneier-samsung-tv-listening/>.

Schneier B, 2016, "Data is a toxic asset", *CNN*, 1 March, <http://edition.cnn.com/2016/03/01/opinions/data-is-a-toxic-asset-opinion-schneier/>.

Thrift N, French S, 2002, "The automatic production of space", *Transactions of the Institute of British Geographers*, 27(3): 309-35.

van Kranenburg R, 2008 *The Internet of Things: A Critique of Ambient Technology and the All-seeing Network of RFID Network Notebooks*, <http://networkcultures.org/wpmu/portal/publications/network-notebooks/the-internet-of-things/>.

Williamson B, 2015, "Algorithmic skin: health-tracking technologies, personal analytics and the biopedagogies of digitized health", *Sport, Education and Society*, 20(1): 133-51.

Zetter K, 2016, "Inside the cunning, unprecedented hack of Ukraine's power grid", *Wired News*, 3 March, www.wired.com/2016/03/inside-cunning-unprecedented-hack-ukraines-power-grid.

Ziewitz M 2016, "Governing algorithms: myth, mess, and methods", *Science, Technology, & Human Values*, 41(1): 3-16.

Zook M, Graham M 2007, "The creative reconstruction of the internet: Google and the privatization of cyberspace and digiplace", *Geoforum*, 38: 1322-43.

Figure 1 (a, b)



Figure 2 (a, b)

18/7/89
 Kilburn Highest Factor Routerie (amended)-

button	C	26	27	line	0	1	2	3	4	5	13	14	15
-26 to C	-G ₁	-	-	1	0	0	0	1	1	0	1	0	1
c to 26			-G ₁	2	0	1	0	1	1	1	1	0	1
-26 to C	G ₁			3	0	1	0	1	1	1	0	1	0
c to 27			-G ₁	4	1	1	0	1	1	1	1	0	1
-23 to C	a	T _n	-G _n	5	1	1	1	0	1	0	1	0	1
Subr. 27	a	G _n		6	1	1	0	1	1	0	0	1	1
Subr. 26	T _n			7	-	-	-	-	-	0	1	1	1
c to 25				8	0	0	1	0	1	0	1	0	1
-25 to C				9	0	1	0	1	1	1	0	0	1
Subr. 26	T _n			10	1	0	0	1	1	1	1	0	1
c to 25				11	1	0	0	1	1	1	0	0	1
Subr. 26				12	-	-	-	-	-	-	0	1	1
Subr. 21	G _n	T _n	-G _n	13	0	1	0	1	1	1	1	1	1
c to 27	G _{n+1}			14	0	1	0	1	1	1	0	1	0
-27 to C	G _{n+1}			15	1	0	1	0	1	0	0	1	1
c to 26	G _{n+1}			16	1	1	0	1	1	1	0	1	1
22 to 26	T _n	-G _{n+1}	G _{n+1}	17	1	1	0	1	1	1	0	1	1
				18	0	1	0	1	1	1	0	1	1
				19	0	1	1	0	1	1	0	0	0

20	-3	1011 etc	23	-a	25	init. final
21	1	10000	24	G ₁	26	-G _n (0)
22	4	00100			27	-G _n

or 10100

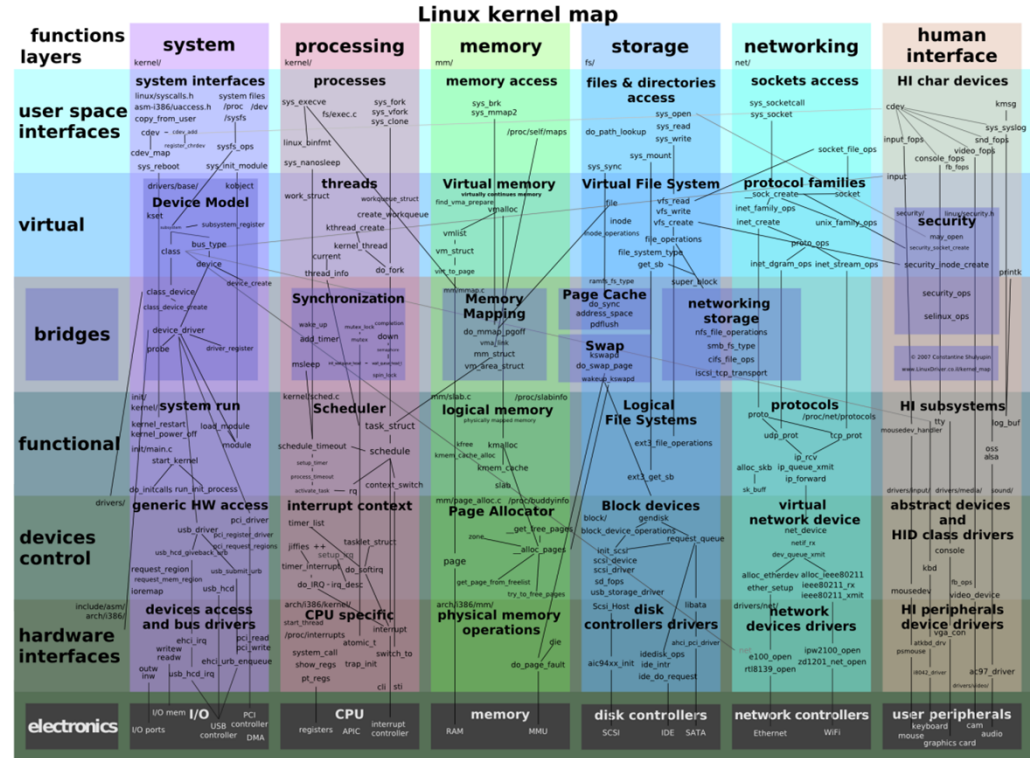


Figure 3

Windows Task Manager

File Options View Help

Applications Processes Services Performance Networking Users

Image Name	PID	Session ID	CPU	CPU Time	Working ...	Peak Working S...	Memory...	Base Pri	Threads	I/O Reads	Description
firefox.exe *32	5244	1	00	04:13:49	641,108 K	1,538,424 K	480,940 K	Normal	56	295,623	Firefox
POWERPNT.EXE *32	5116	1	00	00:00:21	93,308 K	162,756 K	64,396 K	Normal	10	735	Microsoft PowerPoint
explorer.exe	1372	1	00	00:16:54	93,068 K	107,732 K	62,412 K	Normal	41	529,751	Windows Explorer
wmplayer.exe *32	5908	1	02	00:02:04	64,192 K	65,976 K	37,676 K	Normal	21	2,165	Windows Media Player
WINWORD.EXE *32	7200	1	00	00:00:18	71,160 K	75,500 K	24,244 K	Normal	11	5,725	Microsoft Word
dwm.exe	2820	1	00	00:36:10	27,684 K	36,568 K	14,064 K	High	5	1	Desktop Window Manager
i_view32.exe *32	7288	1	00	00:00:00	18,164 K	20,196 K	12,648 K	Normal	1	90	IrfanView
WINWORD.EXE *32	6032	1	00	00:00:06	9,060 K	72,672 K	4,284 K	Normal	10	1,067	Microsoft Word
PfUsSmon.exe *32	944	1	00	00:00:03	6,948 K	33,332 K	4,148 K	Normal	6	4,274	ScanSnap Manager
taskhost.exe	4448	1	00	00:00:05	7,684 K	13,516 K	3,372 K	Normal	11	132	Host Process for Windows Tasks
taskmgr.exe	5896	1	01	00:00:03	11,780 K	12,036 K	3,216 K	High	6	2	Windows Task Manager
csrss.exe	888	1	00	00:02:54	92,884 K	101,544 K	1,820 K	High	12	1,564,617	
updateui.exe *32	5772	1	00	00:00:02	3,276 K	25,416 K	1,740 K	Normal	4	1	updateui.exe
splwow64.exe	3096	1	00	00:00:00	3,488 K	6,648 K	1,516 K	Normal	9	1	Print driver host for 32bit applications
igfxpers.exe	4804	1	00	00:00:00	3,028 K	8,412 K	1,056 K	Normal	5	0	persistence Module
ETDCtrl.exe	4224	1	00	00:00:11	1,808 K	17,420 K	1,008 K	Above ...	10	1	ETD Control Center
ismagent.exe *32	6204	1	00	00:00:13	1,688 K	28,156 K	864 K	Normal	80	642	Intel Services Manager
RAVCpl64.exe	5108	1	00	00:00:00	1,776 K	12,224 K	860 K	Normal	9	0	Realtek HD Audio Manager
notepad.exe	6232	1	00	00:00:03	3,028 K	7,880 K	836 K	Normal	1	2	Notepad
SsWiaChecker.exe *32	3476	1	00	00:00:00	1,764 K	5,520 K	836 K	Normal	5	1	ScanSnap WIA Service Checker
winlogon.exe	932	1	00	00:00:00	2,088 K	9,316 K	764 K	High	4	4	
rundll32.exe	4028	1	00	00:00:00	1,716 K	10,120 K	744 K	Normal	3	131	Windows host process (Rundll32)
notepad.exe	8088	1	00	00:00:00	2,344 K	7,620 K	700 K	Normal	1	2	Notepad
taskeng.exe	4496	1	00	00:00:00	1,476 K	7,300 K	692 K	Normal	5	14	Task Scheduler Engine
notepad.exe	4240	1	00	00:00:01	2,228 K	7,468 K	664 K	Normal	1	2	Notepad
VCDDaemon.exe *32	4456	1	00	00:00:00	1,632 K	5,624 K	660 K	Normal	3	1	Virtual CloneDrive Daemon
hkcmd.exe	4784	1	00	00:00:00	1,228 K	7,388 K	584 K	Normal	3	0	hkcmd Module
igfxtray.exe	832	1	00	00:00:00	1,212 K	7,592 K	576 K	Normal	3	0	igfxTray Module
btplayerctrl.exe *32	5724	1	00	00:00:00	976 K	6,280 K	552 K	Normal	4	1	Bluetooth Media Player Controller
iusb3mon.exe *32	1380	1	00	00:00:01	1,112 K	5,876 K	544 K	Normal	4	1	Intel(R) USB 3.0 Monitor
ALMon.exe *32	4248	1	00	00:00:01	916 K	8,028 K	524 K	Normal	13	442	Sophos Endpoint Security and Control
AdobeARM.exe *32	4644	1	00	00:00:01	988 K	249,040 K	492 K	Normal	6	230	Adobe Reader and Acrobat Manager
SSFoldTray.exe *32	5712	1	00	00:00:00	712 K	5,352 K	356 K	Normal	1	1	SSFoldTray
BleServicesCtrl.exe	4708	1	00	00:00:00	728 K	8,204 K	352 K	Normal	5	1	Bluetooth LE Services Control Program

Show processes from all users

Processes: 101 CPU Usage: 3% Physical Memory: 46%

Figure 4



Figure 5 (a, b)



Programmed to make a mess

Programmed to clean up

iRobot Roomba: cleans routinely...so you don't have to

The new floor vacuuming robot that really works

Keeping on top of the cleaning and floor cleaning in particular is a constant battle in any home. Thankfully, the new iRobot Roomba is designed to relieve you of this tedious chore and help you on a daily basis. It cleans floors superbly at the press of a button, using less energy than a standard vacuum.

How does iRobot Roomba work? Advanced sensors and AWARE® robot technology ensure this intelligent and energy efficient home robot covers your whole floor area. Whether it's carpets, rugs or hard floor surfaces, its highly effective brush system and smart vacuum picks up large debris as well as fine dust and dirt. It even gets right under most furniture to clean those difficult to reach areas.

Just turn it on, walk away and come back to clean, mess free floors.

For more information on iRobot Roomba call 0800 132 509

Distributed by **Domotec** home appliances

2007 iRobot Corporation. All rights reserved. iRobot and Roomba are registered trademarks of iRobot Corporation.

www.irobot.com

iRobot®



Navigates to clean right under low furniture

Adjusts to clean any type of floor surface

Cleans along edges and in tight corners

Automatically returns to homebase to recharge

The idea.



Special offer!
£1119.99
20% OFF
SAVE £280
all manufacturers published RRP of £1,399.99 (inc VAT)

The finished product.
 Designed for that constantly perfect green lawn without the effort.



WIN HUSQVARNA PRODUCTS
 visit husqvarna.com/uk for details

THE NEW AUTOMOWER® 305

Why mow the lawn when the new Automower® 305 can mow it for you? This new robotic lawnmower automatically cuts the grass all on its own. You simply leave it to get on with the job. When it's finished, it returns to its charging station without you having to do anything, allowing you to enjoy your garden and your leisure time. It cuts the grass little and often, returning clippings into the lawn to provide a natural fertiliser. The result is a greener, healthier-looking lawn with that 'fresh cut' look. No wonder so many people are flocking to take advantage of our special introductory offer.

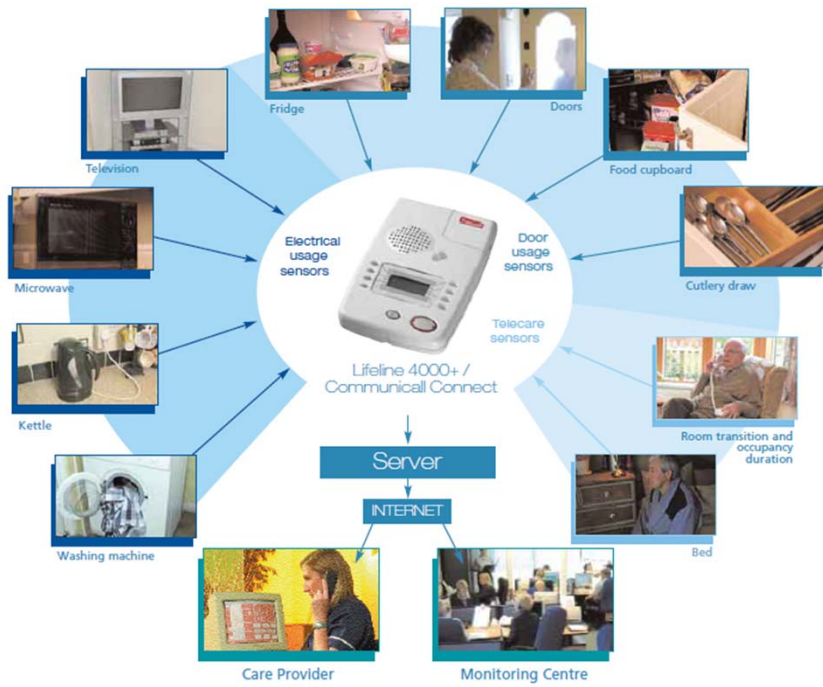


For more information (Brochure), details of current promotional offers or to find your nearest stockist visit husqvarna.com/uk or call us on 0800 169 6148.

*Offer ends 30th June 2012 and is subject to availability of goods, while promotional stocks last. Available at participating dealers only.
 © 2012 Husqvarna AB (pub). All rights reserved. Husqvarna and other product and feature marks are trademarks of Husqvarna AB (pub). Terms and conditions apply.



Figure 6 (a, b)



Activity monitoring

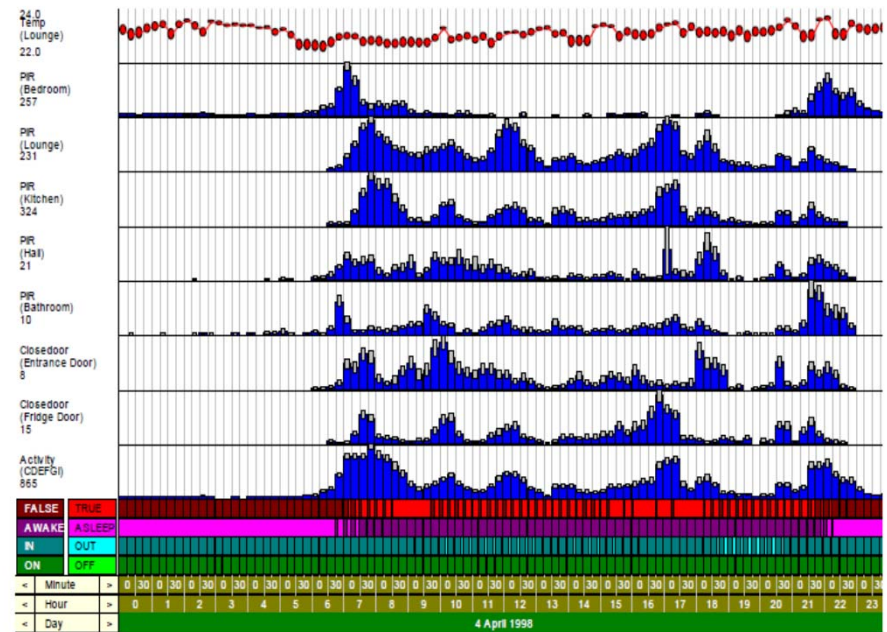
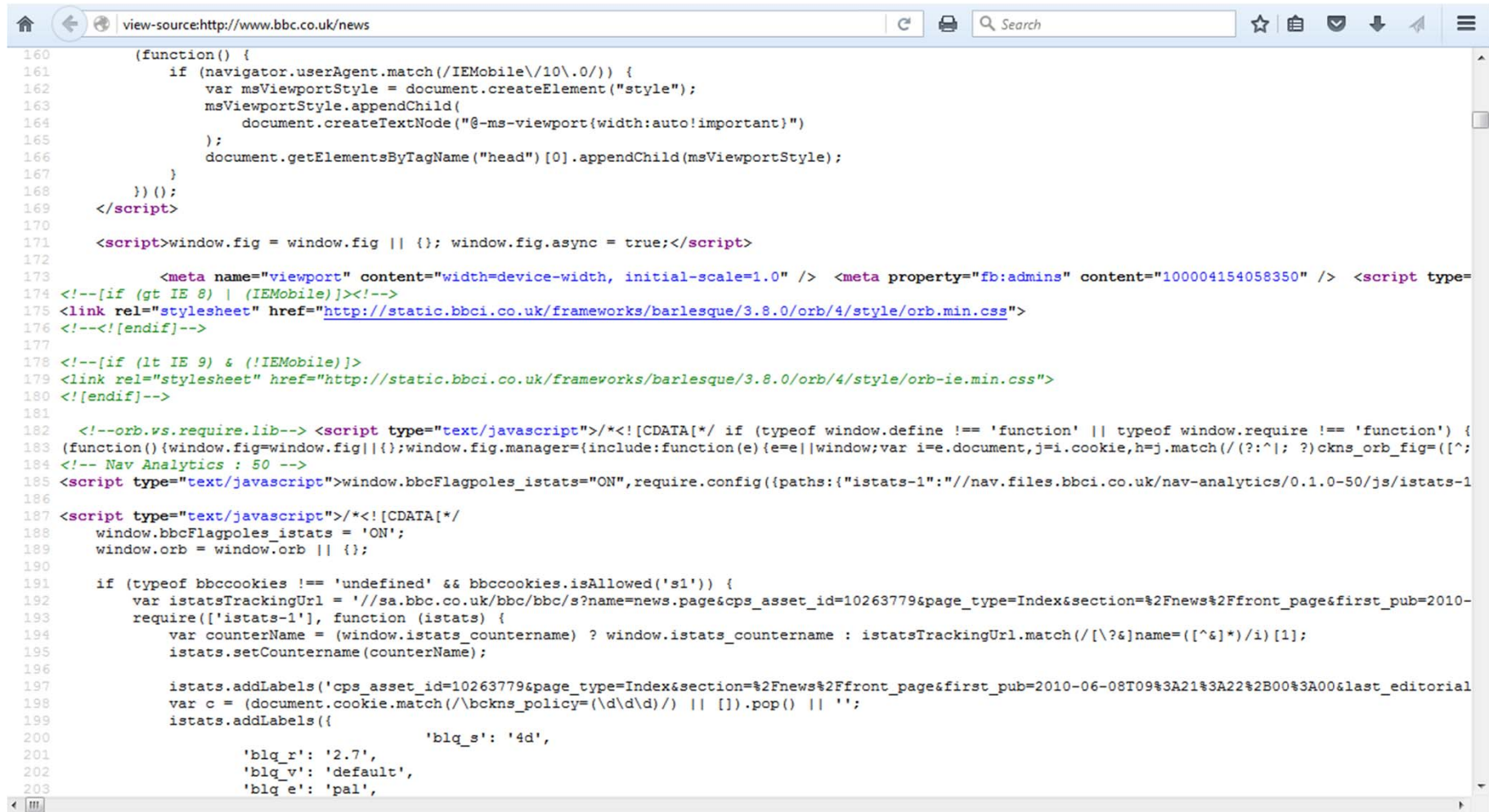


Figure 7 (a, b)



Figure 8



```
160     (function() {
161         if (navigator.userAgent.match(/IEMobile\/10\.0/)) {
162             var msViewportStyle = document.createElement("style");
163             msViewportStyle.appendChild(
164                 document.createTextNode("@-ms-viewport{width:auto!important}")
165             );
166             document.getElementsByTagName("head")[0].appendChild(msViewportStyle);
167         }
168     })();
169 </script>
170
171 <script>window.fig = window.fig || {}; window.fig.async = true;</script>
172
173     <meta name="viewport" content="width=device-width, initial-scale=1.0" /> <meta property="fb:admins" content="100004154058350" /> <script type=
174 <!--[if (gt IE 8) | (IEMobile)]><!-->
175 <link rel="stylesheet" href="http://static.bbc.co.uk/frameworks/barlesque/3.8.0/orb/4/style/orb.min.css">
176 <!--![endif]-->
177
178 <!--[if (lt IE 9) & (!IEMobile)]>
179 <link rel="stylesheet" href="http://static.bbc.co.uk/frameworks/barlesque/3.8.0/orb/4/style/orb-ie.min.css">
180 <![endif]-->
181
182 <!--orb.ws.require.lib--> <script type="text/javascript">*<![CDATA[*] if (typeof window.define !== 'function' || typeof window.require !== 'function') {
183 (function(){window.fig=window.fig||{};window.fig.manager={include:function(e){e=e||window;var i=e.document,j=i.cookie,h=j.match(/(?:^|; ?)ckns_orb_fig=(?:^
184 <!-- Nav Analytics : 50 -->
185 <script type="text/javascript">window.bbcFlagpoles_istats="ON",require.config({paths:{"istats-1":"//nav.files.bbc.co.uk/nav-analytics/0.1.0-50/js/istats-1
186
187 <script type="text/javascript">*<![CDATA[*]
188 window.bbcFlagpoles_istats = 'ON';
189 window.orb = window.orb || {};
190
191 if (typeof bbccookies !== 'undefined' && bbccookies.isAllowed('s1')) {
192     var istatsTrackingUrl = '//sa.bbc.co.uk/bbc/bbc/s?name=news.page&cps_asset_id=10263779&page_type=Index&section=%2Fnews%2Ffront_page&first_pub=2010-
193     require(['istats-1'], function (istats) {
194         var counterName = (window.istats_countername) ? window.istats_countername : istatsTrackingUrl.match(/[\?&]name=(?:^&)*\/i)[1];
195         istats.setCountername(counterName);
196
197         istats.addLabels('cps_asset_id=10263779&page_type=Index&section=%2Fnews%2Ffront_page&first_pub=2010-06-08T09%3A21%3A22%2B00%3A00&last_editorial
198         var c = (document.cookie.match(/\bckns_policy=(?:\d\d\d\d\/) || []).pop() || '');
199         istats.addLabels({
200             'blq_s': '4d',
201             'blq_r': '2.7',
202             'blq_v': 'default',
203             'blq_e': 'pal',
```