Programming power: Policy networks and the pedagogies of 'learning to code'

Ben Williamson, University of Stirling

Chapter abstract

'Learning to code' has transformed from a grassroots movement into a major policy agenda in education policy in England. This chapter provides a 'policy network analysis' tracing the governmental, business and civil society actors now operating in 'policy networks' to mobilize learning to code in the reformed National Curriculum. Learning to code provides evidence of how power over the education policy process is being displaced to cross-sector actors such as 'policy labs' that can broker networks across public and private sector borderlines. It also examines how the pedagogies of learning to code are intended to inculcate young people into the material practices and ways of seeing, thinking and doing associated with the professional culture of programmers, the emerging context of solutions-engineering in social and public policy, and with the participatory culture of social media 'prosumption.'

Interest in the educational value of learning to write and programme computer code has grown from a minority concern among computing educators, grassroots computing organizations, and computer scientists into a major policy discourse. Originating with activist and grassroots campaigning groups such as Computing at School, 'learning to code' is now being actively promoted in England by cross-sector organizations including Nesta (National Endowment for Science, Technology and the Arts) and the Nominet Trust that are increasingly seeking to participate in educational governance, as detailed later. As a result, learning to code has been recognized as desirable amongst politicians and educational policymakers, as evidenced by the scheduled 2014 replacement in the English National Curriculum of ICT (Information and Communication Technology)which emphasizes office skills, with a new computing programme of study which emphasizes computer science and programming skills (Department for Education 2012). Learning to code has been transformed from a grassroots campaign into a major policy agenda in a remarkably concentrated period, yet the powerful actors mobilizing it into curriculum policy are largely unrecognized in educational policy research, and the material practices of

coding promoted through the pedagogies of learning to code have not been subject to detailed research.

Learning to code is additionally embedded in a contemporary societal context in which software codes and algorithms are understood as increasingly powerful influences on the world. The rapid expansion of 'Big Data,' 'machine learning' and 'data analytics' reflect a contemporary situation in which software code and algorithms are being put to work as powerful technologies right across social, political, cultural and economic contexts and governmental, civil society and industrial sectors, as well as in science, social science, and humanities disciplines (Kitchin 2014). A form of technical 'solutionism' is emerging across sectors, fields and disciplines which tends to assume that all social, scientific, governmental and human problems can be addressed through the application of the right code and algorithms (Morozov 2013). As a consequence, a critical social scientific debate (largely among sociologists, geographers and 'software studies' researchers, e.g. Fuller 2008) has developed around code, algorithms, and software. Researchers increasingly recognize software code and algorithms as an 'invisible structural force' that can 'pattern and coordinate everyday life' (Mackenzie 2006: 45). Terms such as 'algorithmic power,' 'code as law,' and 'algorithmic ideology' have proliferated (e.g. Mager 2012). As the title of a new book by media theorist Lev Manovich (2013) asserts, *Software Takes Command*. He argues that the contemporary world has undergone a transformational 'softwarization' into a 'software society' in which all social, economic, and cultural systems of modern society now run on software and its constitution through code (Manovich 2013). Like electricity and combustion in the industrial society, he claims, software enables global information society. To date, little critical attention has been given to software code or digital data in educational research, though, as Selwyn (2014: 9) notes, there is now increased emphasis on 'the "modelling" of education through digital data' and 'algorithmically-driven "systems thinking"—where complex (and unsolvable) social problems associated with education can be seen as complex (but solvable) statistical problems.' Moreover, Williamson (2014a: 2) has identified how the 'algorithmic power' of 'network-based and database-driven software' is 'increasingly augmenting, mediating and governing educational practices.' These accounts suggest that software code has been increasingly positioned as the solution to educational problems. It is in this context that a variety of organizations and actors has coalesced around learning to code, although this is not a coherent and stable network but a messy hybrid of intentions, ambitions, and interests.

In this chapter, I trace the policy developments, discourses, and cross-sectoral and interorganizational connections that have translated learning to code into curriculum policy. The chapter is organized around two clusters of questions. The first cluster of questions is about power and policy networks. What organizations are involved in seeking to influence and negotiate policy around learning to code? Is this an example of how power in the educational policy making process is being displaced to new networks of actors? The second cluster of questions is about the power of software itself as an actor in education. With computer code and programming activities increasingly prominent, are we seeing the emergence of new nonhuman sources and configurations of power? How might we understand the power of computer coded devices themselves? Can these influence what learners do? And how do the pedagogies of learning to code configure and activate the capacities of the learner? These are questions central to the aim of this book: to explore new actors and agents of power in education, and to explore new forms of power operating in different contexts. The chapter combines aspects of policy studies with software studies approaches in the social sciences to consider the power of learning to code in education.

Policy network analysis

The chapter draws on a study of the participation in education of cross-sector organizations, think tanks, and other 'policy intermediaries' and 'policy labs.' The focus is specifically on the organizations Nesta and the Nominet Trust, and on the ways that they have established networks of governmental, civil society and commercial actors to promote and campaign for learning to code. Nesta was established as the National Endowment for Science, Technology and the Arts by the UK New Labour government in 1998 but became independent in 2011 with a remit to innovate in public services; 'digital education' is one of its key themes and the platform on which it advocates a range of learning to code initiatives. Nesta's activities around learning to code are all managed within its 'Public Innovation Lab' which seeks to solve social challenges through the application of new technologies. The Nominet Trust was established in 2008 by Nominet, the internet registry which maintains the .uk register of domain names. The Nominet Trust invests in projects and programmes 'using the internet to address big social challenges,' and describes itself through the discourse of social investment, social innovation, and social technology entrepreneurship. The Nominet Trust hosts the 'Social Tech Guide' website which showcases technology projects which 'address complex social

challenges, from health and education to poverty and climate change.' It positions technology as a 'social good' and its steering committee consists of both the chief executive of the Nominet Trust and the chief executive of Nesta. In collaboration, Nesta and Nominet Trust are major partners in the campaign Make Things Do Stuff which promotes a wide range of activities and organizations associated with learning to code and other forms of 'digital making.'

Organizationally, these organizations are neither governmental nor commercial actors, but straddle sectors and broker projects and connections between them. Nesta and the Nominet Trust both act as 'hubs' for a variety of partnerships and networks. They are prototypical of 'public and social innovation labs' ('psilabs') or 'i-teams' (innovation teams) as Nesta documents describe them (Mulgan 2014; Nesta 2014). Public and social policy innovation labs seek to put 'smart software' and digital data to work deep within the activities of government, alongside new forms of 'sociable governance' through relationships and collaborations, particularly in the redesign of public services, education, health, and social services (Williamson 2014b). Nesta's own public innovation lab and the Nominet Trust's emphasis on social innovation and 'social tech' are evidence of how such organizational reconfigurations are enabling them to position themselves as solution-providers for public and social policy problems. Policy labs, or psilabs and i-teams, are a new organizational format combining a variety of 'sociable' methods of co-design, rapid prototyping, design ethnography, and citizen entrepreneurship with 'smart' coded methods, such as data mining, data analytics, and predictive 'machine learning' methods in the redesign of services such as education. In this emerging sector, code and algorithms are seen as engineering solutions to the problems of re-engineering government, as 'hack events' sponsored by Nesta, such as government hacking and 'hackathons' for public sector redesign, clearly demonstrate (Merrett 2014).

These organizations are contributing to new forms of cross-sectoral 'network governance' and 'policy networks' in public education in England (Williamson 2014c). 'Networked governance' is characterized by decentralization, mobility, fluidity, looseness, complexity and instability, by the criss-crossing of sectoral borderlines and the hybridization of ideas, discourses and materials from bureaucratic, academic and media fields. Educational 'policy networks' are a specific interorganizational materialization of network governance. Made up primarily of 'experts' from think tanks, policy institutes, multilateral agencies, media consultancies, political lobbying and public relations, policy networks 'perform the

role of conveying ideas between different areas of the production, distribution, or circulation of ideas' in order to 'influence the decision-making process' (Lawn & Grek 2012: 75). While the concepts of network governance and policy networks are not uncontentious, Ball and Junemann (2012) claim that in England education policy certainly is now being dispersed and enacted by increasingly heterogeneous and sometimes unstable networks of governmental, civil society and business actors.

In seeking to demonstrate how education is increasingly being governed through network governance, and through associated organizational configurations of 'policy labs' and 'social innovators,' this chapter is focused on how intermediary policy actors are promoting the practices of learning to code in schools. Learning to code is both a set of pedagogic practices and a contemporary policy discourse being enacted by a mixture of actors from policy labs, governmental agencies, and commercial companies, through a variety of projects, partnerships and campaigns. Through such networks, learning to code is being constructed as a hybrid product of different discourses, interests and agendas. Adopting methods of 'policy network analysis,' I focus on the reports, pamphlets, websites and other documents which articulate these intermediaries' ideas and aspirations. As Ball & Junemann (2012: 14) articulate it, the method of policy network analysis seeks to identify actors, their associations and relationships, and their power and capacities to contribute to policy decision-making. The specific focus below is on identifying key organizations from government, business and civil society involved in promoting various activities around 'learning to code' (primarily in England), and on analyzing the ways in which they discursively construct and mobilize learning to code.

The central argument is that intermediary organizations such as Nesta and the Nominet Trust are promoting computer programming activities in ways which embed young people firmly in the coded infrastructures and material practices of today's digitally-mediated landscape. This demands a consideration of how power is being displaced both to intermediary actors and to the coded infrastructures and programming practices they promote. In the next section, I seek to understand 'code' as an increasingly pervasive source of power in the world, before proceeding to examine the formation of the 'learning to code' agenda.

Programming power

Computer code is commonly understood as the machine-readable language programmed to instruct computer software. It is the substrate to software, and is

constructed through programming—the art and science of putting together algorithms and instructions that can be automatically read and translated by a machine in order to do something. A growing recognition of the power of code is reflected in popular science publications like *9 Algorithms that Changed the Future* (MacCormick 2013) which demonstrate the vast reach of code and its algorithmic ordering structures into contemporary everyday practices. The algorithms of the title refer to search engine indexing and ranking; the cryptographic algorithms required by secure websites; pattern recognition algorithms for recognising handwriting, speech and faces; data compression of files like MP3s and JPEGs; and the transactional changes made to databases, such as those required for online banking and social networking sites like Facebook. All of these algorithms and their rules and sequences are written in code, making code itself into a significant contemporary material device, the coders that script it into significant actors, and the coding they do into a significant material practice.

Beyond its technical and material existence, code also exerts important social effects. Computer code is thoroughly entangled in contemporary practices of surveillance, enterprise, consumption, leisure, economics, politics, and much else, as developments such as government snooping, 'smart cities,' personalized targeted advertising, and the transformation of online popular culture attest (Mackenzie 2006; Beer 2013). As code is wired out into the world in software products, it is now understood among many social scientists as more than just the written script that instructs and controls computing devices. As Manovich (2013: 15) phrases it, software is '*a layer that permeates all areas of contemporary societies'* (*original italics*). Through the software it instructs, code organizes, disrupts and participates in contemporary social, economic, political and cultural activities and practices. It may even be 'reassembling social science' itself (Ruppert, Law & Savage 2013) as new digital methods and search algorithms make possible new analyses, configurations and visualizations of the world. Sociotechnically understood as both a product of the world and a relational *producer* of the world, code *acts*: it interpolates, mixes with and ultimately *produces* collective political, economic and cultural life (Kitchin & Dodge 2011). It is inseparable from its social, cultural, political and economic processes of *production* and its socially, culturally, politically and economically *productive* effects.

Moreover, people view and understand code through the deployment of powerful and consistent discourses that promote, justify and naturalize software across a

whole array of domains (Kitchin & Dodge 2011). Indeed, for Mackenzie and Vurdubakis (2011: 16), software is the 'hybrid progeny of computer code and social codes of conduct': not just the technical fact of lines of code that instruct software, but sets of social codes with the power to 'direct how citizens act' (Thrift 2005: 173). All of these things add up to a pervasive system of thought within which the procedures and processes written in code and sequenced in algorithms may be taken as a new set of rules and mundane routines to live by. The power of code is not just in its technical operations but in how it sinks into everyday cultural, economic and political discourse, thought and action.

We need to consider here the idea that code acts as a 'vital source of social power' that augments society (Kitchin & Dodge 2011: 246). As the substrate to software, code is significant as a source of power because it can 'make things happen' by virtue of its 'execute-ability,' its ability to perform tasks according to encoded instructions (Mackenzie & Vurdubakis 2011: 6). Software code is not inert but fundamentally performative. The performativity of code to make things happen and to produce outcomes autonomously lies at the heart of many recent accounts of the role of software in modern life, to the extent that some researchers consider software code and algorithms as a challenge to human agency itself. As Beer (2009: 987) claims, 'algorithmic power' may be 'becoming a part of how we live, a part of our being, a part of how we do things, the way we are treated, the things we encounter, our way of life.'

Scott Lash (2007) has described the power of software code in a technologically mediated world as 'power after hegemony.' His article is an ambitious reconsideration of cultural theory; here I want to pick up on the major point he raises about power and algorithms. Lash (2007: 55-56) argues that in cultural studies 'hegemony means domination through consent as much as coercion,' through ideology and discourse, and 'that cultural power is largely addressed to the reproduction of economy, society and polity.' For Lash, our new era, however, is thoroughly technologically mediated and consequently things like computer code and its algorithms are introducing their 'rules' into human societies. In contrast to the reproductive logic of hegemony, in a new epoch 'post-hegemonic power operates through a logic of invention, hence not of reproduction but of chronic production of economic, social and political relations' (Lash 2007: 56). These rules are 'generative' and 'inventive', and as algorithms increasingly pervade the social fabric as new kinds of social rules, they therefore have the generative and inventive capacity to

shape and configure social formations and individual lives. This is because code, by its very nature, changes things: it transforms an input into an output; and because algorithms function by ordering, structuring and sequencing things (Mackenzie 2006). As such, how code and algorithms are programmed can extend into the ordering, structuring and sequencing of the social world itself. In this sense, we are moving into a 'society in which power is increasingly in the algorithm' (Lash 2007: 71); where power is in the software we use and where, as Beer (2009: 995) adds, 'information is harvested about us' in order to generate new experiences. Thus, we move in a world where software 'learns' about us:

> This is undoubtedly an expression of power, not of someone having power over someone else, but of the software making choices and connections in complex and unpredictable ways in order to shape the everyday experiences of the user. (Beer 2009: 997)

Whereas hegemonic power sought to secure consent through ideology and discourse to the reproduction of economy, society and polity, a post-hegemonic form of algorithmic power generates new configurations of social, economic and political practice. Algorithmic power constantly generates new realities. This is the case, for example, when Amazon's algorithms generate recommendations for consumer purchases; when Google's PageRank algorithm orders search query results; or when Facebook's NewsFeed algorithm configures users' social network feeds; but even more significantly when algorithmic data analytics systems automate such things as the provision of government services, organize transport and utilities infrastructures, coordinate social control mechanisms, and enable real-time governmental and commercial surveillance. In all these cases, the generative rules of algorithms work from inside everyday life rather than from outside in the form of a dominant ideology or discourse. They constitute 'grammars of action' for new forms of social ordering and governance, and are endowed with the power to 'actively reshape behaviour' (Kitchin & Dodge 2011: 109). As Beer (2013: 70) explains, 'algorithms create realities, they constitute the world in different ways and they present us with limitations and boundaries that we then live by.' As such, Mager (2012) has argued that the kind of code and algorithms that facilitate everyday social practices such as using search engines are based on particular social models of the world. In order for an algorithmic system to function, Neyland (2014) claims, the world outside of the system has to be mathematically modelled in such a way that it can be built-in to the social world of the algorithmic system. Google's driverless car, for example, relies on ultra-precise digitized maps to navigate the physical world—a compelling case of the ways in algorithms and their code are involved in building 'a

world out there into a world in here, in the algorithmic machine' (Neyland 2014: 11). Ideological hegemonic power from outside is being elided by post-hegemonic power that, through the ubiquity of devices programmed in code and algorithms, acts as a 'technological unconscious' patterning everyday activity (Thrift 2005).

Software code and algorithms are, then, through and through social *products* as well as *producers* of the world. They model the world in particular ways, and provide grammars of action which make it possible then to act on that world. In the next sections I explore how this understanding of code might enable us to better understand emerging developments around programming and learning to code in schools. Learning to code inculcates young people in the material practices of code production, whilst also embedding them firmly in a heavily software-mediated environment structured and ordered by code and algorithmic power.

Programming pedagogies

Today there is a growing interest in promoting computer programming to young people. In this section I explore the ways in which 'learning to code' has been discursively constructed and promoted by cross-sector intermediary organizations, including Nesta and the Nominet Trust, which act as conduits for a network of interests from the governmental, commercial and civil society sectors. As we shall see, the result is that 'learning to code' has become a contingent, hybrid and elastic concept. As the *Observer* newspaper columnist Naughton (2012) has stated, the growth of interest in programming comes with a variety of different assumptions from advocates across different sectors. The dominant policy interest, he argues, is in promoting and growing computer entrepreneurship for the economic competitiveness of 'UK plc.' The alternative perspective, which Naughton advocates, is learning to code for *informed citizenship* in a world where computation has become ubiquitous. Naughton draws specifically on a catchy slogan from Douglas Rushkoff (2010), who states that 'if you are not a programmer, you are one of the programmed,' and argues that learning to code is essential if we wish to understand how our technologies work and how they work on us. These arguments certainly appear to acknowledge that the world is increasingly governed by coded products, and suggest that 'learning to code' is a way of giving learners some agency to counteract its pervasive power.

The evidence that such arguments for learning to programme computer code has been taken up in educational discourse is in the fast growth of 'Code Club', a

volunteer-based grassroots initiative that places computer programmers in after-school clubs in primary schools to teach young children basic programming and coding. According to the organizers of Code Club:

> Learning to code is an important skill now we're living in a digital age. It's not just enough for children to know how to use technology. They should know how it works too. … They should understand that they're in charge of the computer, and can (and should) make it do what they want, not the other way around. (Code Club 2013)

Code Club was established in April 2012 and has quickly grown into a nationwide network of clubs (at the time of writing in summer 2014) in over 2000 UK primary schools, as well as a Code Club World network. Code Club is sponsored and promoted by Nesta and the Nominet Trust with funding from the Department for Education, as well as support from computing corporations like Microsoft and Google. It is marketed simultaneously in terms of its educational benefits and the economic benefits of upskilling children as computer programmers. The organizers of Code Club are extremely active on social media such as Google Groups and Twitter, where they coordinate many activities such as 'Code Pub' meet-ups for volunteers, have been profiled frequently by the press, and appear at many events including practical workshops and conferences. Code Club is not just a set of educational activities but a whole culture of programming, including participants from infancy up to the professional programming domain, materialized in practices ranging from basic coding tutorials and games to high level advanced programming.

Code Club is one among many grassroots initiatives that, during 2013, were increasingly clustered and networked together as part of a concerted campaign to promote young people learning to code. In May 2013, the Public Innovation Lab at Nesta, in partnership with the not-for-profit 'social innovator' the Nominet Trust and the internet company Mozilla, launched an initiative called Make Things Do Stuff that promotes various forms of learning to code, programming and 'digital making':

> Make Things Do Stuff aims to mobilize the next generation of digital makers. We want to help people to make the shift from consuming digital technologies, to making and building their own. Because when all kinds of different people start hacking, re-mixing and making things with technology, the possibilities get really interesting. Make Things Do Stuff will enable people to … navigate a path that will take them from being a digital consumer, to being a digital maker. (Make Things Do Stuff 2013a)

These activities are justified through a combination of discourses about the powerful role of computer code in the contemporary world and the need of commercial

computer companies. The Make Things Do Stuff website states that: 'In a world where everything from fridges to cars, bank accounts to medical diagnoses are becoming powered by computing, understanding how digital technologies are made (and how to make your own) is vital to full participation in society' (Make Things Do Stuff 2013b). Furthermore, it juxtaposes a constructionist understanding of 'making something, sharing it and getting feedback' as ' a powerful way to learn,' with a commercial discourse of how 'digital technologies are developed in the real world: get something made, get it out there, get feedback, learn, and make it better' (Make Things Do Stuff 2013b). The Make Things Do Stuff campaign is the hybrid progeny of educational, governmental, commercial, and grassroots discourses and the actors and organizations that actively promote them.

Make Things Do Stuff is primarily organized and governed by its three major partners, Nesta, the Nominet Trust and Mozilla, though its activities are distributed among a wide cross-sectoral network of government, civil society and commercial actors. As a source of funding, support, and campaigning, Make Things Do Stuff has distributed funding and support to Code Club as well as a number of related coding and 'digital making' activities such as CoderDojo clubs in Scotland and Technocamps in Wales. The initiative is described as an 'open movement' and is partnered with a range of technology companies, education businesses, third sector organizations, and government. These include Facebook, Microsoft, O2, Mozilla, and Virgin Media; Codecademy, Coding for Kids, Decoded; and HM Government, the Scottish Government and the Teacher Development Trust. The government Chancellor of the Exchequer, George Osborne MP, launched the initiative in May 2013 claiming that 'this campaign is backing the entrepreneurs of the future and helping ensure that Britain is equipped to succeed in the global race' (HM Treasury 2013).

There is a clear cross-sectoral policy narrative around programming as an economically valuable skill in evidence here. To give some more detail to this narrative, these entanglements of computer companies with government via intermediaries such as Nesta and the Nominet Trust have influenced the scheduled 2014 replacement of the 'ICT' with 'computing' in the National Curriculum in England. The computing programmes of study explicitly focus on programming and coding along with 'computational thinking' and core knowledge from computer science (Department for Education 2013). The impetus to replace ICT with computing in the curriculum was led by a Royal Society (2012) report *Shut Down or*

Restart which was directly commissioned by Microsoft, Google, and university computer science departments, and the new computing curriculum has been developed by the British Computer Society and the Royal Academy of Engineering with leadership from a Microsoft senior executive (Harrison 2012).

The Cambridge academic and Observer newspaper columnist John Naughton (2012) has contributed to the debate with a high profile series of articles including a 'Manifesto' for reintroducing computer science in schools. Naughton uses the expression 'program or be programmed,' the title of the book by Douglas Rushkoff (2010), who also works as an adviser for Codecademy, itself a Make Things Do Stuff partner organization. Rushkoff's book itself has spawned a number of online 'study guides' which aim to make its key ideas accessible to a much wider and younger audience. There has also been considerable grassroots support for programming in the curriculum from Computing at School, a member-led subject association for computing teachers, which is chaired by a senior Microsoft researcher and is funded by Microsoft, Google and the Chartered Institute of IT. The Computing at School 'white paper' of 2010 was among the first documents to argue for the replacement of ICT in the National Curriculum with computing. The paper from Computing at School (2010) argued that 'computing is the study of how computers and computer systems work, and how they are constructed and programmed,' and it suggested that a new computing curriculum would include the study of 'how computers work,' how algorithms, data structures, systems and networks are used to solve computational problems, as well as teaching the knowledge and skills of programming. This is largely the message of the new computing curriculum itself, and the DfE has subsequently awarded funding (alongside Microsoft, Google and others) for Computing at School to support a 'Network of Teaching Excellence in Computer Science' to grow teaching capacity in advance of its implementation (Computing at School 2014).

However, it was only in 2011 when Nesta published a report entitled *Next Gen* (Livingstone & Hope 2011) that the key messages about computing and learning to code took on policy significance. *Next Gen* demanded more 'rigorous teaching of computing in schools' and recommended putting computer science into the national curriculum. The report did not originate, however, from a concern with the teaching of computing in schools. Rather, it was commissioned as a review of the skills needs of the videogames and visual effects industries, which have long been seen as economically valuable and innovative sectors of the UK economy. The authors are

industry leaders in the videogames and visual effects sector and the report was commissioned by Ed Vaizey, the Conservative Party Minister for Culture, Communications and the Creative Industries. The importance of *Next Gen* was signalled after Eric Schmidt from Google used the platform of the MacTaggart Lecture at the Edinburgh television festival in 2011 to express his dismay that computer science was not taught as standard in UK schools, a message repeated by Google around the world urging governments to support young people to learn to code in order to produce a skilled workforce for a digital economy (Cave & Rowell 2014).

As a partner in Make Things Do Stuff, the Nominet Trust, too, has produced a series of reports, events, projects and blogs dedicated to the topic of learning to code. Echoing political discourse on the subject, the Nominet Trust chief executive Annika Small claims there is a 'serious and economic imperative' besides the 'fun and learning that digital making offers young people,' namely that the 'UK and global jobs market are crying out for digital skills and we need to make sure that the next generation can meet this need' (Nominet Trust 2013). Nominet Trust has distributed funding through a 'Digital Makers Fund' in partnership with Nesta. The beneficiaries include a number of start-up organizations and grassroots organizations involved in various digital making and learning to code activities. Nominet Trust also commissions reports and 'state of the art' reviews on key areas such as digital making, big data, and the politics of computers (e.g. Sefton-Green 2013; Krotoski 2014). It represents a messy mix of advocacy for the digital economy, support for grassroots organizations, the social economy and civil society, as well as journalistic and academic commentary on aspects of digital culture, within which its campaigning for learning to code is entangled.

Make Things Do Stuff, Code Club and related activities in the UK have been mirrored at an international scale. In the US, during 2013, a campaign called 'Hour of Code' was launched which called for all school children to learn some programming skills, based on a clear argument about the economic benefit of equipping young people for jobs in computer science related jobs. Promotional material produced early 2014 claims the campaign reached 20million young people in December 2013 alone, and aims to involve over 100 million in 2014 through a mix of online courses, tutorials and video lectures made available to schools (code.org 2014). In terms of governance, Hour of Code was set up and run by code.org, 'a non-profit dedicated to expanding participation in computer science by making it

available in more schools.' Its 'vision is that every student in every school should have the opportunity to learn computer programming' (code.org 2014). A non-profit organization, code.org was founded by the entrepreneurs Ali and Hadi Partovi, twins with a long history of 'angel investment' and venture capitalism in Silicon Valley, has been partnered with or sponsored by donations from Microsoft, Google, Amazon, Dropbox, Facebook, and many others, as well as by philanthropic individuals from across commercial computing and venture capitalism (see https://code.org/about for a full and extensive list of organizational and individual partners and donors).

Back in the UK, a similar campaign was launched in January 2014. The 'Year of Code' was established to coincide with the introduction of the new computing curriculum in England which puts coding in the curriculum for every schoolchild, and is an active campaign to promote a variety of programming and coding initiatives both in and out of school, to help people 'learn code & create exciting things on computers' (Year of Code 2014). The Year of Code website (http://yearofcode.org/) provides links to a range of start-up organizations and grassroots campaigns related to learning to code, as well as an extensive network of partners from across government, commercial media, and civil society. Year of Code is chaired by Rohan Silva, a former senior policy advisor to prime Minister David Cameron, and an 'entrepreneur-in-residence' at Index Ventures, an international venture capital firm whose mission statement is that 'every aspect of human life and economic activity can be transformed by technology and entrepreneurial passion' (http://indexventures.com/firm). Its executive director and advisors are almost all drawn from the fields of entrepreneurship, venture capital, and computing. Its only explicitly educational advisor is from the Education Foundation, an 'independent think tank for education' that advocates and champions digital innovation in education and acts as a partner with other technology companies, notably Facebook, to introduce their products in schools. . As the Guardian columnist John Naughton (2014) argued, 'Year of Code is a takeover bid by a corporate world that has woken up to the realization that the changes in the computing curriculum … will open up massive commercial opportunities.' The BBC journalist Rory Cellan-Jones (2014) revealed that one of its founders, Saul Klein, also of Index Ventures, when pushed to discuss whether Year of Code was a government or Index Ventures initiative, claimed that: 'We live in a world where the intersection of public policy and commerce is often needed to drive an important social agenda.' The development of Hour of Code in the US and Year of Code in the UK is evidence of how initial

grassroots movements and activities, such as Computing at School and Code Club, have been mediated by increasingly powerful cross-sectoral policy innovation labs and absorbed into the entrepreneurial mission of venture capital companies.

For the commercial sector, there may be clear economic benefits to be gained from supporting learning to code. As Morozov (2014) has written, the 'learning to code,' educational 'hacking' and the 'maker movement' are all highly desirable to some of the most powerful agencies and organizations in the world. Google, Facebook, and Microsoft have all supported high profile campaigns like the Hour of Code, while DARPA (the defence research wing of the American military complex) has spent over $13million promoting the maker movement and 'makerspaces' to high schoolers, and in China the Communist Youth League has been actively recruiting participants to 'Maker Carnivals.' The desirability of such activities is most obviously in the upskilling of a future workforce, as many advocates for learning to code demonstrate. In their book on political lobbying in the UK, Cave and Rowell (2014: 260-61) describe the various activities surrounding the learning to code movement and the reform of the computing curriculum as a 'lobbying tool for technology firms with a clear, vested interest in digitizing learning, as well as enthusing a new generation of coders.' They claim that this campaign of 'business-backed think tanks' and 'education technology lobbyists' 'intent on reshaping education' (249) has now 'got what it wanted' in the shape of computer science in the curriculum, twinned with much great political acceptance by the Department for Education of technology being 'integrated and embedded across the whole curriculum' and its desire to build a strong UK educational technology market (261). Beyond general arguments about upskilling for the digital economy or growing the educational technology market, learning to code is also embedded in concerns about the capacity of businesses and government agencies to make use of Big Data sources and more intelligent, connected devices, as outlined in a report by the government Design Commission (2014) which recommends further governmental support for the teaching of code in the curriculum as well as digital making and shared 'makerspaces/hackspaces' in schools, colleges and universities.

Learning to code is no longer simply an after-school activity run by volunteer programmers, as originally envisioned by Code Club and other likeminded grassroots organizations such as Computing at School. As these sets of entanglements between government, businesses, intermediaries, lobbyists, and educational organizations demonstrate, learning to code has become the focus for

the development of complex new cross-sectoral alignments and networks. Year of Code and Make Things Do Stuff exemplify the kind of cross-sectoral 'policy networks' that are increasingly participating in educational governance in England, especially around new technologies, Big Data and new media agendas. The learning to code policy domain and its discourse is not merely a government product, but the hybrid and ultimately messy result of pronouncements produced by computing specialists, entrepreneurs and investors, journalists, policymakers, lobbyists, and corporate computing companies, brokered by intermediary partnering organizations and policy innovation labs such as Nesta and the Nominet Trust. It demonstrates clearly how educational governance is increasingly being displaced to powerful new actors from outside of the educational sector itself, influenced by powerful interests and ambitions, with complex links between governmental, business and civil society organizations and practices.

Programmers and prosumers

Despite its rapid growth, the underlying assumptions about learning to code have gone largely unquestioned. Clearly coding carries into the classroom a specific set of assumptions about knowledge and forms of knowing and doing. As noted earlier, programming is not just a technical procedure but is related to systems of thought about the way the world works, and about how it might be modelled in order to further shape people's interactions with it. Indeed, for Kitchin & Dodge (2011: 26) the material practice of programming is 'an expression of how the world can be captured, represented, processed and modelled computationally with the outcome subsequently doing work in the world.' For example, the ways in which the world of banking can be captured in online banking systems, or how biometric systems are constructed to facilitate automated border control, subsequently shape how these activities take place. In other words, programming code captures ideas about how the world works, in order to then augment, mediate and regulate people's lives. Though, as Kirschenbaum (2009) has pointed out, any act of programming may contain biased, distorted, caricatured, or merely partial selections from the world it claims to model; in that sense, programming is a persuasive or perhaps rhetorical act. The material practice of programming, therefore, possesses the power to shape how people know and act in the social world.

Moreover, material practices of learning to code assume a certain image of the desirable individual learner to be produced. As the researcher of code cultures Mackenzie (2006) argues, the work of computer programmers is premised on

notions of flexibility, speed, virtuality, just-in-time-production, teamwork, and other aspects of 'immaterial labour.' Make Things Do Stuff, Code Club, Year of Code and the like anticipate learners' entry into a network-based digital economy for which the work of programmers stands as a prototypical practice. Thus an emphasis on learning to code is part of what Barry (2001) describes as the contemporary political preoccupation with sculpting a mind and body with the technical skills, knowledge and capacity to meet the demands of new flexible work routines. Consistent with much recent education policy discourse, learning to code activities 'govern by activating the capacities of the individual' to contribute to the digital economy (Ozga, Segerholm & Simola 2011: 88). In this sense, learning to code may be interpreted as a material practice of 'algorithmic ideology' (Mager 2012), a kind of inculcation into the codes of conduct, practices, assumptions, and knowledges that underpin production in the digital economy. Thus, learning to code embodies a host of assumptions and working practices based on ideas such as computational thinking, statistical modelling, systems thinking, scientific rationality, and algorithmic logic that have their origins in the working practices of the computing professions. These are very particular kinds of social practices imbued with 'particular values and contextualized within a particular scientific approach,' and often reductionist, functionalist and technicist modes of thinking that see the world in computational terms rather than in relation to cultural, economic or political context (Kitchin 2014: 5). To adapt Lash's (2007: 75) terms, what are being rehearsed through learning to code are the 'hands-on' practices and epistemologies of 'coders, writing algorithms,' working in 'ephemeral project-networks' in 'laboratories and studios.' In a culture where power is in the algorithm, Lash argues, status goes to those actors with the material skills, social values, and expert epistemologies to construct those algorithms. At its most basic, such practices amount to the fantasy of technical 'solutionism' where the right code and algorithms may be seen as the solution to complex problems. Learning to code thus seeks to inculcate learners into the systems of thought associated with programmers, and with the knowledge and philosophies of the world, with all their biases, prejudices, ideological assumptions and modes of perception, that are materialized in software products.

It is clear that for its advocates at Nominet Trust and Nesta, as well as both the governmental and business actors with which they are networked, that coding is positioned as a rewarding, desirable and skilled occupation, not least in terms of providing technical engineering solutions to public and social policy problems. Both Nesta and Nominet Trust support 'hack' events such as 'government hacking' and

'hackathons' which put teams of computer programmers together, using code-sharing tools, to engineer solutions to intractable government and public sector problems (Merrett 2014). The Nominet Trust's 'Social Tech Guide' provides ample evidence of how technology entrepreneurship, twinned with practices of coding and hacking, has been positioned for 'social good' (Nominet Trust 2014). According to Morozov (2013) this kind of solutionist thinking originates in the Silicon Valley hacker culture of technological innovation, which has recast complex social phenomena like politics, public health, and education, as neatly defined problems with definite, computable solutions that can be optimized if the right algorithms are in place. The overall digital making, coding and hacking discourse is embedded in this social, cultural and political context of technological utopianism. Via Nesta and the Nominet Trust, and through their networks and associations with the culture of hack events, learning to code has been positioned in relation to such activities as equipping young people with the skills required to become solutions-engineers and hackers of the future.

Yet the depiction of solutionist hacking glosses over the fragility, complexity and mundanity of much coding work in the digital economy. As Mackenzie (2006: 14) notes, software has to be coded, and yet this job may be undertaken by 'a programmer, webmaster, corporation, software engineer, team, hacker or scripter…. The figure of the programmer often vacillates between potent creator of new worlds and antisocial, perhaps criminal or parasitic.' More prosaically, the work and material practice of coding is often dull, routinized and monotonous, as well as difficult, frustrating and dysfunctional (Kitchin & Dodge 2011). Moreover, as Kitchin and Dodge (2011: 33) have argued, coding is a 'disciplinary regime' with established 'ways of knowing and doing regarding coding practices.' Yet owing to intense ongoing innovation in the field, programmers are always struggling to learn and adapt to constant change and experience a high degree of 'ignorant expertise' and confusion about what they are doing (Ullman cited in Kitchin & Dodge 2011: 35), particularly in relation to the wider possible social effects of what is incorporated into the code. Coders simply do not always know the effects of the code they are writing, and nor do they acknowledge how their own worldviews, ideologies and assumptions are embedded in the kinds of interactions and forms of doing that they make possible. The frequent failure of software projects, the 'bitrot' that occurs as software packages are constantly superseded, and the regular disruptions caused by software bugs in everything from online banking to password protection are all evidence of the fragility and contingency of the code produced through the material

practices of coders. Moreover, the construction of software features which breach people's right to privacy indicate how many coding projects proceed without amply considering their wider social effects, as evidenced by recent European Union proposals over the 'right to be forgotten' which favour the rights of individuals rather than software companies to manage and control their personal data. The learner participating in Code Club, Year of Code, Make Things Do Stuff, or the like, is being solicited into a system of thought, ways of seeing, knowing and doing associated with a culture of coding practice that is not always as systematic, objective and expert as it is widely represented as being by learning to code advocates. The material practice of coding is more complex, contingent, confused, ignorant, and distanced from concerns over its effects on the social world, and rests on the assumption that the problems with that social world can be addressed with algorithmic solutions written in code. This is about applying technical engineering to the task of human and social engineering. Learning to code is premised on a fantasy of the material practices associated with coding which simplifies and romanticizes the empirical reality of disciplinary practice in the digital economy.

However, Make Things Do Stuff and Code Club justify themselves not just through the prospective economic value of children learning to code, but through a wider cultural argument about *producing* and not simply *consuming* technology. One way to analyze this preoccupation with coding clubs, programming and related digital making activities is to view it as promoting 'participatory' practices of 'co-production,' 'crowdsourcing' and 'prosumption' in new social media practices. 'Prosumption' registers the alleged blurring of production and consumption as consumers of digital media increasingly also become its producers. Manovich (2013: 18-19), for example, argues that 'software development is gradually getting more democratized' as a result of the recent simplification of programming environments through social media. The argument that software production, coding, and other forms of prosumption are ultimately democratizing and empowering has been taken up enthusiastically by Code Club in particular, and also repeated by both the Nominet Trust and Nesta, albeit as part of a messy mix of commercial, economic and civil society discourses and arguments.

From a more critical perspective, Beer & Burrows (2013) question the apparent 'democratization' of software, claiming that this logic plays back into the hands of commercial digital media organizations. They argue that network-based social

media—Facebook, Twitter, YouTube, Wikipedia, and so on—have facilitated the increasing participation of people in the formation of media content, leading to the:

> significant phenomena of the growing amount of 'labouring' people are undertaking as they 'play' with these new technologies: creating profiles, making status updates; distributing information; sharing files; uploading images; blogging, tweeting; and the rest. (Beer & Burrows 2013: 49)

Ideas associated with participation in the networked cultures of social media, such as co-production, prosumption, crowdsourcing, user-centred design, and so on, have long been attractive with organizations such as Nesta, which has put such practices at the centre of its reformatory ambitions for 'digital education' as well as more widely in its proposals for 'people-powered' public services and new 'conversational' forms of 'sociable governance' (Williamson 2014a; 2014b). Learning to code is a logical outgrowth of this proliferation of technologies of co-construction, crowdsourcing and prosumption.

However, network-based activities of programming, prosumption and so on are also interweaving individuals more and more densely into new data-based social media infrastructures. In their analysis of social media in contemporary popular culture, Beer and Burrows (2013) argue that data accumulation does not just 'capture' culture but is recombined through feedback loops to actually shape, reconstitute and co-construct popular culture and everyday practices. They offer examples such as automated recommendations services and 'behavioural advertising' in consumption practices (techniques commonly practised by Amazon, Google, Spotify, Facebook and other social media services). These services accumulate personal and behavioural data from online transactions and run these data through predictive analytics in order to generate personalized recommendations. On the basis of users' subsequent behaviour, these systems then work recombinantly and recursively by continually harvesting users' by-product data and feeding it back into their predictive recommendations.

Through learning to code, young people are increasingly being positioned as 'prosumers' whose active production of online content—in the shape of Facebook updates, tweets, online purchases, and so on—is now the basis for the business models of most major social media companies. The job of the prosumer is to produce content from which commercial organizations can attempt to extract value. Moreover, these data can then be used to modify future services and

recommendations—thus subtly and continually reshaping cultural engagement itself. In this sense, learning to code firmly embeds young people in what Kitchin and Dodge (2011: 6) term the 'coded infrastructures' that now orchestrate many of the patterns of everyday life, and that and are subject to the commercial interests of for-profit communication corporations. Consequently, learning to code is not a neutral or depoliticized material practice, but shaped, patterned, ordered and governed by powerfully commercialized coded infrastructures. In turn, through their material participation in the coded infrastructures of prosumption, young people are being shaped and moulded with particular ways of seeing, thinking, and acting; their digital subjectivities sculpted by the systems of thought programmed into the software they use. The prosumerist individual configured by the software of social media providers is encouraged to share personal information and data; maximize sociality through horizontal networks of connected friends; extend reach, influence and collaboration through liking and sharing digital artefacts; and to contribute through everyday participative and creative forms of digital making, software programming, and coding. Learning to code is a material practice that takes place in the coded infrastructures of contemporary algorithmic power.

Conclusion

This chapter has begun to unpack two emerging issues in terms of power in education. Through documenting and analyzing the recent growth of learning to code in schools, it has shown, firstly, how education is increasingly being targeted by intermediary organizations which represent particular kinds of agendas and concentrate a variety of powerful interests from across the commercial, civil society and governmental sectors. Education policy is being discussed and made in new places, by new actors, through new forms of network governance and through relationships among policy networks. The planned introduction of the new computing curriculum in England in 2014, with its strong emphasis on computer science, computational thinking and computer programming over ICT skills, demonstrates how the networking together of commercial and governmental interests, much of it accomplished through relationships brokered by intermediary organizations such as Nesta and the Nominet Trust and by their discursive production of reports and campaigns, is now exerting considerable influence on mainstream educational policymaking. As a policy discourse, not just a set of pedagogies, learning to code is evidence of shifting power relations in education policy and governance. Specifically it is evidence of the displacement of power to cross-sector intermediaries such as public and social policy innovation labs, and of

their capacity to broker connections, conversations and new forms of 'sociable governance' among distributed 'policy networks' of governmental, civil society and commercial actors.

Secondly, the chapter has shown how, through the work of these intermediary actors, education is increasingly being embedded in coded infrastructures which demand a reshaping of young people's capacities and abilities. Through learning to code, young people are being inculcated into the material practices and codes of conduct associated with the cultures, ways of viewing the world, and ideologies of computer programmers—particularly the assumption that technical engineering, algorithms and coding solutions can be applied for 'social good' and to 'hack' human, social and public problems. As producers and not just consumers of coded products, they are also being embedded as prosumers in the infrastructures of contemporary social media participation, making their everyday activities amenable to the extraction of value by powerful commercial social media companies and to the subtly recursive shaping of contemporary life.

The learning to code movement has been transformed from its origins among grassroots movements such as Computing at School. It has become the focus for a variety of powerful commercial, governmental and civil society actors, mediated by intermediaries and venture capital organizations that are little recognized in educational research. While some of its original enthusiasts and advocates saw learning to code as a way to give power back to users, or to stimulate informed citizenship for an increasingly digitally dense world, it has been translated into the business model of global social media and computing corporations, mobilized in political ambitions for a digital economy, and embedded as a material practice of prosumption in the coded infrastructures of a recursive digital culture. This is a culture in which, as Lash (2007) argues, power is in the algorithm—where software and its code and algorithms are constantly generating new realities, and where young people are being configured in the conduct of coders, with the skills and capacities to write the software and algorithms that will engineer, activate and 'hack' the future.

References

Ball, S. J. & Junemann, C. 2012. *Networks, New Governance and Education*. Bristol: Policy Press.

Barry, A. 2001. *Political Machines: Governing a technological society.* London: Athlone Press.

Beer, D. 2009. Power through the algorithm? Participatory web cultures and the technological unconscious. *New Media and Society* 11, no. 6: 985-1002.

Beer, D. 2013. *Popular Culture and New Media: The Politics of Circulation*. London: Palgrave Macmillan.

Beer, D. & Burrows, R. 2013. Popular culture, digital archives and the new social life of data. *Theory, Culture and Society* 30, no. 4: 47-71.

Cellan-Jones, R. 2014. Year of Code—PR fiasco or vital mission? *BBC online*, 12 February: http://www.bbc.co.uk/news/technology-26150717 (accessed 28 April 2014)

Code.org. 2014. Overview. Code.org website. Available online: https://code.org/files/Code.orgOverview.pdf (accessed 28 April 2014)

Code Club. 2013. About Code Club. *Code Club* website. Available online: https://www.codeclub.org.uk/about (accessed 13 September 2013).

Computing at School. 2010. Computing at School white paper. *Computing at School* website, available online: http://www.computingatschool.org.uk/data/uploads/Computing_at_School.pdf (accessed 16 June 2014).

Computing at School. 2012. Network of Teaching Excellence in Computer Science. *Computing at School* website, available online: http://www.computingatschool.org.uk/index.php?id=noe (accessed 16 June 2014).

Department for Education [DfE]. 2012. 'Harmful' ICT curriculum set to be dropped to make way for rigorous computer science. *Gov.uk* website, available online: https://www.gov.uk/government/news/harmful-ict-curriculum-set-to-be-dropped-to-make-way-for-rigorous-computer-science (accessed 13 September 2013).

Department for Education [DfE]. 2014. National Curriculum in England: Computing programmes of study. *Gov.uk* website, available online: https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study (accessed 16 June 2014).

Design Commission (2014). *Designing the Digital Economy: Embedding growth through design, innovation and technology*. London: Design Commission. Available online: http://www.socialtech.org.uk/about/ (accessed 16 June 2014).

Fuller, M. (ed.) 2008. *Software Studies: A lexicon*. London: MIT Press.

HM Treasury. 2013. 100,000 young people to become 'digital makers.' Gov.co.uk. Available online: https://www.gov.uk/government/news/100000-young-people-to-become-digital-makers (accessed 13 September 2013).

Kirschenbaum, M. 2009. Hello worlds! The Chronicle of Higher Education, 23 January. Available online: http://chronicle.com/article/Hello-Worlds/5476 (accessed 13 September 2013).

Kitchin, R. 2014. Big Data, new epistemologies and paradigm shifts. Big Data & Society, April-June 2014: 1-12. DOI: http://dx.doi.org/10.1177/2053951714528481

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

Krotoski, A. 2014. *The personal (computer) is political*. Oxford: Nominet Trust. Available online: http://www.nominettrust.org.uk/knowledge-centre/articles/personal-computer-political (accessed 25 April 2014)

Lash, S. 2007. Power after Hegemony: Cultural studies in mutation? *Theory, Culture & Society* 24, no. 3: 55-78.

Lawn, M. & Grek, S. 2012. *Europeanizing Education: Governing a New Policy Space*. Oxford: Symposium.

Livingstone, I. & Hope, A. 2011. *Next Gen*. London: Nesta.

Mackenzie, A. 2006. *Cutting Code: Software and Sociality*. Oxford: Peter Lang.

Mackenzie, A. & Vurdubakis, T. 2011. Codes and Codings in Crisis: Signification, Performativity and Excess. *Theory, Culture and Society* 28, no. 6: 3-23.

MacCormick, J. 2012. *9 Algorithms that Transformed the Future: The Ingenious Ideas that Drive Today's Computers.* Oxford: Princeton University Press.

Mager, A. 2012. Algorithmic ideology: How capitalist society shapes search engines. *Information, Communication & Society* 15, no. 5: 769-787.

Make Things Do Stuff. 2013a. About Make Things Do Stuff. *Make Things Do Stuff* website. Available online: http://makethingsdostuff.co.uk/about (accessed 13 September 2013).

Make Things Do Stuff. 2013b. What is Make Things Do Stuff? *Make Things Do Stuff* blog. Available online: http://makethingsdostuff.co.uk/blog/2013/03/12/what-make-things-do-stuff (accessed 13 September 2013).

Manovich, L. 2013. *Software Takes Command*. London: Bloomsbury.

Merrett, N. 2014. Public sector embraces 'hackathon' innovation potential. *Government Computing*, 30 May. Available online: http://central-government.governmentcomputing.com/news/public-sector-embraces-hackathon-innovation-potential-4281766 (accessed 16 June 2014).

Morozov, E. 2013. The perils of perfection. *New York Times,* Sunday Review, 2 March. Available online: http://www.nytimes.com/2013/03/03/opinion/sunday/the-perils-of-perfection.html (accessed 16 June 2014).

Morozov, E. 2014. Making it. *The New Yorker*, 13 January. Available online: http://www.newyorker.com/arts/critics/atlarge/2014/01/13/140113crat_atlarge_morozov (accessed 16 June 2014).

Mulgan, G. 2014. The radical's dilemma: an overview of the practice and prospects of Social and Public Labs—version 1. Available online: http://www.nesta.org.uk/sites/default/files/social_and_public_labs_-_and_the_radicals_dilemma.pdf (accessed 26 February 2014).

Naughton, J. 2012. A manifesto for teaching computer science in the 21st century. *The Observer*, 31 March. Available online: http://www.theguardian.com/education/2012/mar/31/manifesto-teaching-ict-education-minister (accessed 13 September 2013).

Naughton, J. 2014. Year of Code already needs a reboot. *The Guardian*, 15 February: http://www.theguardian.com/technology/2014/feb/15/year-of-code-needs-reboot-teachers (accessed 28 April 2014).

Nesta. 2014. i-teams. Nesta website. Available online: http://www.nesta.org.uk/project/i-teams (accessed 16 June 2014).

Neyland, D. 2014. On organizing algorithms. *Theory, Culture & Society*, DOI: http://dx.doi.org/10.1177/0263276414530477

Nominet Trust. 2013. Digital making activities to expand opportunities for UK young people. Nominet Trust website. Available online: http://www.nominettrust.org.uk/news-events/news/digital-making-activities-to-expand-opportunities-uk-young-people (accessed 13 September 2013).

Nominet Trust. 2014. About the Social Tech Guide. *Social Tech Guide* website. Available online: http://www.socialtech.org.uk/about/ (accessed 16 June 2014).

Ozga, J., Segerholm, C. & Simola, H. 2011. The governance turn. In Ozga, J., Dahler-Larsen, P., Segerholm, C. & Simola, H. (eds). *Fabricating Quality in Education: Data and governance in Europe*, 85-95. London: Rouledge.

Ruppert, E., Law, J. & Savage, M. 2013. Reassembling social science methods: The challenge of digital devices. *Theory, Culture and Society* 30, no. 4: 22-46.

Rushkoff, D. 2011. *Program or Be Programmed: Ten Commands for the Digital Age*. ORbooks.

Sefton-Green, J. 2013. Mapping Digital Makers. Oxford: Nominet Trust. Available online: http://www.nominettrust.org.uk/knowledge-centre/articles/mapping-digital-makers (accessed 16 June 2014).

Selwyn, N. 2014. Data entry: toward the critical study of digital data in education. *Learning, Media & Technology*, DOI: http://dx.doi.org/10.1080/17439884.2014.921628

Thrift, N. 2005. *Knowing Capitalism*. London: Sage.

Williamson, B. 2014a. Governing software: networks, databases and algorithmic power in the digital governance of public education. *Learning, Media & Technology*, DOI: http://dx.doi.org/10.1080/17439884.2014.924527

Williamson, B. 2014b. Policy labs, sociable governance and smart software. *Code Acts in Education* blog, 2 April. Avauilable online: http://codeactsineducation.wordpress.com/2014/04/02/policy-labs-sociable-governance-and-smart-software/ (accessed 16 June 2014).

Williamson, B. 2014c. Mediating education policy: Making up the 'anti-politics' of third sector participation in public education. *British Journal of Education Studies* 62, no. 1: 37-55.

Year of Code. 2014. What is Year of Code? *Year of Code* website, available online: http://yearofcode.org/ 9accessed 16 June 2014).