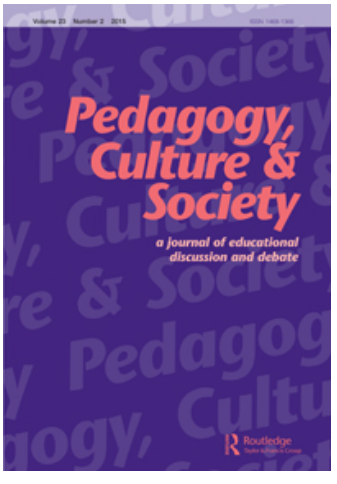

Pedagogy, Culture & Society
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/rpcs20

Software and the hidden curriculum in digital education

Richard Edwards[a]
[a] School of Education, University of Stirling, Stirling, UK
Published online: 12 Dec 2014.

To cite this article: Richard Edwards (2015) Software and the hidden curriculum in digital education, Pedagogy, Culture & Society, 23:2, 265-279, DOI: 10.1080/14681366.2014.977809

To link to this article: http://dx.doi.org/10.1080/14681366.2014.977809

Pedagogy, Culture & Society, 2015
Vol. 23, No. 2, 265–279, http://dx.doi.org/10.1080/14681366.2014.977809

Software and the hidden curriculum in digital education

Richard Edwards*

School of Education, University of Stirling, Stirling, UK

Computer technologies and computer-mediated information and communication are increasingly parts of curriculum-making practices in education. These technologies are often taken to be simply tools to be used to enhance teaching and learning. However, in recent years, a range of cross-disciplinary studies have started to point to the work of code, algorithms and standards in selecting and shaping the information, forms of knowledge and modes of interaction available to teachers and students. Concerns have been raised about how data is selected, shaped and represented by software in ways which are not always apparent to those using computer technologies. In this sense, software can be considered as part of the hidden curriculum of education. Drawing upon the increasing research in software studies, this article explores theoretically some of the issues raised in relation to curriculum-making practices and possible lines of empirical research to be pursued.

Keywords: digital education; curriculum; software; code; algorithms; standards

Introduction

Computer technologies and computer-mediated information and communication are increasingly parts of educational practices at all levels in most parts of the globe. These technologies are often taken simply to be tools to be used to enhance teaching and learning or support the more efficient governing and management of institutions through 'big data' (Lawn 2013). Technology has always been important to education, whether it is pens, backboards or immersive simulated environment (e.g. Lawn and Grosvenor 2005), but it is arguable that the speed and scope of innovation in computer technologies is at a faster pace and more pervasive than we have seen previously. This is, in part, as Manovich (2013, 222) argues because 'the general logic of software industry is to always try to offer users "more" than the previous application's version of competitors'. In other words, there is a competitive logic in the computing industry to innovate, and the newness of innovation can itself be seductive.

*Email: r.g.edwards@stir.ac.uk

While such technologies take many forms, their ubiquity is arguably matched by their increasing taken-for-grantedness. The technologies are often black-boxed and naturalised, and left unexamined other than at the level of their use (Bowker and Star 2000). They are there and they do the work we need them to do until of course they breakdown or do something that we do not expect. The black-boxing and taken-for-grantedness is almost deliberate, designed into the technology, in the sense that few understand how things work or can repair them, if and when they break.

How then do we research, frame and theorise the entanglements of computer technologies in education? For a start, clear and agreed definitions of terms, such as hardware, software, networks, algorithms, etc. are hard to establish, not least because each are, by themselves, always already assemblages. The entanglements of technology in work and daily practices are not new research questions and there are different approaches to theorising in this arena as elsewhere (e.g. Orlikowski 2007; Faulkner, Lawson, and Runde 2010, Kitchin and Dodge 2011). There are many attempts to frame computer technologies as acting in the world, while not falling into technological determinism. Examining the uptakes of computer technology on work organisations and practices has been a focus for many years, with particular concerns about the deskilling and reskilling of the workforce, and the resultant struggles over shifting statuses and rewards (e.g. Farrell 2006; Eriksson-Zetterquist, Lindberg, and Styhre 2009). There is also interest in the changing relationships with clients and customers of services, as, for instance, computer technologies enable less face-to-face contact with providers and the increasing use of data analytics to potentially personalise services (Pariser 2011). The market in this data is itself not insignificant. Data analytics is part of a wider trend in relation to the gathering, scraping and mining of data, both in relation to enhancing personal services through big data analysis – in the case of education, through learning analytics (e.g. Ferguson and Buckingham Shum 2012) – and in terms of monitoring and increasing the accountability of workers to ensure effective and efficient services. As Berry (2011, 3) argues therefore, ‘computers run software that is spun like webs, invisibly around us, organising, controlling, monitoring and processing’. There is increasing interest in the ways in which such data is used in the policy domain, including educational policy, through the development of what has been termed ‘governing by data’ (e.g. Lawn 2013).

These are important areas for research. However, what has yet to be fully explored is the ways in which the software that make these technologies operate, assume and produce certain affordances and affects in their development and uptakes. In other words, in examining the role of computer technologies in education, we need to examine the work of code, algorithms and standards in curriculum-making practices in more detail. This becomes even more of an imperative, given moves towards the extension of openness in education represented by initiatives such an open

educational resources, MOOCs, and open access initiatives, all of which rely precisely on the performative work of software. What might be opened in the work of specific forms of software and what might be closed are important questions to be examined.

In recent years, a range of cross-disciplinary studies have started to point to the work of code, algorithms and standards in selecting and shaping the information, forms of knowledge and modes of interaction made available to people. This work is informed by the notion that code is 'technical and social, and material and symbolic simultaneously' (Berry 2011, 36). Code also relies on establishing sets of standards, yet 'because standards are so pervasive that they have become taken for granted in our everyday environment, they may become completely embedded in everyday tools of use' (Star and Lampland 2009, 11). In other words, software plays an integral part in everyday practices, yet its role is often hidden. Drawing upon the increasing research in software studies, this article explores theoretically some of the issues raised in relation to curriculum-making practices, in particular the ways in which software can be considered part of the hidden curriculum, and some possible lines of empirical research to be pursued.

The article is in three parts. First, I will rehearse briefly some of the broader discussion of the hidden curriculum and suggest how the work of software can be considered itself hidden. Second, I will explore more fully the emerging research on the significance of software for curriculum-making practices and the possible increasing inscrutability in evaluating its work and enactments. Third, I will suggest some lines for empirical research in relation to curriculum-making in this area and some of the methodological challenges associated with such an agenda.

The hidden curriculum

Discussion of the hidden curriculum in education is long-standing (e.g. Snyder 1971; Apple and King 1983; Margolis 2001). A Google Scholar search for the term (16 May 2014) identifies over 41,000 references to it. It has been used primarily as part of the critique of educational institutions for reproducing implicitly the unequal opportunities, inequalities and exercises of power in the social order. It is argued that while students may be learning particular subjects and skills at a visible level, at a hidden level, they learn many other things, including the possibilities they have within the existing opportunity structures of education and the wider social order. Institutions, in particular schools, might officially develop the curriculum to support the learning of all students. However, those things which are selected and enacted as part of the formal curriculum provide hidden messages to certain groups and types of students that education is 'not for them'. In this way, it is suggested variously that individuals become socialised into particular roles in the social order, social, cultural and human capital is reproduced

and class, gender, racial and other inequalities in education and society continue (e.g. Bourdieu and Passeron 2000).

For instance, the hidden curriculum has been held to convey the message to working-class students that education is not for them, but rather they should take 'working class jobs' with limited possibilities for social mobility. Similarly, it is suggested that the hidden curriculum conveys to many women that their primary role in the social order is caring for others. From this perspective, the hidden curriculum is one of the primary educational ways through which social inequality is reproduced. Given that the maintenance of social order was one of the primary motivations for the establishment of many national education systems, it is perhaps unsurprising that the attempts to critique this role have formulated various notions of the hidden curriculum.

The reference to 'curriculum' in the term suggests that the hidden curriculum is primarily concerned with the forms of knowledge and authoritative discourses made available in educational institutions. However, it is also recognised that the hidden curriculum may not simply be about knowledge, but also the forms of teacher–student and student–student interaction that are deemed allowable in the curriculum. The same is the case for the resources used in classrooms, the books, artefacts, furniture and of course, learning technologies: all may carry hidden as well as explicit messages. This has informed, for instance, the debates about appropriate or authentic content in texts for 'English as a Second Language' students within the context of globalising processes (Cope and Kalantzis 2000; Snyder 2002). Norms and values are embedded in all of these, which may formally be taken to be universal, but actually differentiate unequally as not all will subscribe to or be encompassed by such universals. Deciphering and the capacity to make one's own meaning is often seen as the redemptive educational strategy in opening up the hidden curriculum to students, wherein code-breaking and code-switching enable the making, rather than the simple reproduction of knowledge. However, the extent to which all is made visible through such practices is open to question.

The concept of the hidden curriculum is therefore very broad. It is essentially one that is used to try and explain the inequality that is reproduced through the ostensibly universal provision of a curriculum, in particular in relation to schooling. It is a way of reproducing social order by providing individuals with messages that they have had their opportunity, but they are only fit for certain purposes, for certain positions in that order. These are often ones that reproduce the types of work and lives that their parents had. However, the very existence of the concept of the hidden curriculum and the practices developed to challenge the inequalities in and reproduced by education demonstrates that it is not as totalising and naturalised as it might seem. The inequalities reproduced through education are themselves positioned as problematic. Nor should we suggest that those practices of

education that challenge the hidden curriculum in support of equity do not themselves have a hidden curriculum. There are those who suggest that the hidden curriculum may not always have negative effects, it can be used to promote social changes for the better as well as the worse (Cotton, Winter, and Bailey 2013). This raises the notion that the hidden curriculum should not in and of itself be critiqued and rejected, but that the legitimacy of the ethics and politics embedded in its different enactments needs more careful calculation. In a sense then, there are always hidden aspects of the curriculum, given that it is always a selection in a particular situation and enactment is always emergent and multiple. The issue is not of whether or not there are hidden aspects of the curriculum, it is more a question of the legitimacy of what is hidden.

It is perhaps unsurprising then to suggest that the work of software in education might be considered as part of the hidden curriculum. As indicated in the introduction, the work of code, algorithms and standards in selecting and representing data and in enabling certain forms of interaction all provide possibilities for hidden messages to be taken up and conveyed. However, neither the hidden curriculum, nor the work of software as part of the hidden curriculum has been of much focus for researchers of digital education. One exception is Anderson (2001, 30, emphasis in original), who refers to three different senses of the notion of the hidden curriculum:

(1) a kind of *indoctrination* that attempts to maintain social privilege;
(2) the subtle effects of the setting in which formal education occurs;
(3) the *unstated* rules for necessary completion of formal education studies.

However, he does not examine software specifically as part of this hidden curriculum. In an earlier article, (Edwards and Carmichael 2012), it was argued that the work of code in the uptake of sematic technologies in education could be examined as an aspect of the hidden curriculum. This challenges those approaches to curriculum and pedagogy wherein computer technologies are considered as simply tools by which the curriculum is 'delivered'. In particular, the article argued that the effects of developing standards, code and algorithms on the representation of data, the forms of teaching and learning that are possible, and the notion of the student and teacher and their roles assumed and enacted (e.g. Woolgar 1991) were part of a 'secret code' of the hidden curriculum (see also Pargman and Palme 2009; Loveless and Williamson 2013). In other words, the work of software, what is assumed in its development and what it can and does do, needs to be researched more explicitly in relation to the development of digital education. What then does the research on software suggest for considering it more explicitly as part of the curriculum-making practices and not simply part of the hidden curriculum?

Software and curriculum-making practices

For many, computers simply work, and the rhetoric of hardware, operating systems and software is that of, for instance, enhanced personal efficiency, innovation and productivity, access to new resources, innovations and the extension of reach and impact (e.g. Nespor 2006). Over time, there has developed a powerful rhetoric which encourages people to view computer technology use as an increasing natural and naturalised part of daily life, seamless and unremarkable (e.g. Thrift 2004; Kitchin and Dodge 2011). More generally, the 'tool' tends to be the dominant metaphor for thinking about computer technology, for example, office tools, web tools and authoring tools. However, in a review of theories of information technology, Orlikowski and Iacono (2001, 131) argued that:

> artefacts are usually made up of a multiplicity of often fragile and fragmentary components, whose interconnections are often partial and provisional and which require bridging, integration, and articulation in order for them to work together. We have a tendency to talk of (technological) artefacts as if they were of a piece – whole, uniform, and unified. For example, we talk about 'the Technology', 'the Internet', 'the Digital Economy', as if these are single, seamless, stable, and the same, every time and everywhere. While such simplifications make it easy to talk about technologies, they also make it difficult to see that such technologies are rarely fully integrated, flawless, and unfailing, and that they can and often do break down, wear down, and shut down.

Such concepts also hide the components that enable the technologies to work.

There is a long history of research exploring the development and deployment of computer technologies, their entanglement with daily practices and the shaping of technologies in particular ways as they are taken up in specific contexts. Here the underpinning assumption is often that 'all relations should be seen as both social and technical' (Law and Bijker 1992, 291). This has been captured in strands of research associated with, for instance, actor networks, mangles of practice, sociotechnical ensembles, material sociology, relational materialism and human–computer interaction (Orlikowski 2007). The research interest in software as part of these assemblages has been growing in recent years, as the work of code and algorithms and the infrastructure of standards to support that work have spread and become more ubiquitous. For instance, Thrift (2004) has argued that software has become a 'technological unconscious' of contemporary life. This is because, as he suggests (2005, 240),

> … software has grown from a small thicket of mechanical writing to a forest of code covering much of the globe … code runs all manner of everyday devices, from electric toothbrushes to microwave ovens, from traffic lights to cars, from mobile phones to the most sophisticated computers.

To do this requires enacting information infrastructures, and integral to this are the development, adoption and application of code, algorithms and standards that enable the organising, mobility and representation of data. Yet, as I have indicated, exploring the work of software as an aspect of curriculum-making practices has been a surprisingly small part of educational research.

Berry (2011, 2) argues that

(1) software allows the delegation of mental processes of high sophistication into computational systems. This instils a greater degree of agency into the technical devices;
(2) networked software encourages a communicative environment of rapidly changing feedback mechanisms that ties humans and non-humans together into new aggregates;
(3) there is a greater use of embedded and quasi-visible technologies, leading to a rapid growth in the amount of quantification that is taking place.

Delegation, aggregation and quantification are, therefore, increasingly at play in the uptake of computer technologies. Further, growing research on the work of code and algorithms (e.g. Barocas, Hood, and Ziewitz 2013; Manovich 2013) and code/spaces (e.g. Kitchin and Dodge 2011) in daily life have the potential for providing significant resources through which educators and researchers can explore the active work of software in curriculum-making. This research points us towards giving far more attention to the software and associated practices of computing through which curriculum-making is enacted. It is the codes, algorithms and the linking of data; the applications of technical standards; and ways in which decision-making and reasoning are articulated in software that, along with the hardware and the electronic and electrical infrastructures of networks, make things, like search engines, web applications, visualisation tools, remote-sensing systems, perform in particular ways and become actors in curriculum-making practices. When utilising digital visualisations of data in educational contexts, this raises questions about the quality and reliability of what is drawn upon and what is coded out as well as what is coded in to make the data visible. There is also the question of the semiotic power of the visualisation over the raw data. To have an interactive visualisation of the mapping of, for instance, volcanic eruptions over time is very different from a simple list of places and dates. At one level, such a visualisation might be more engaging – the hidden work of software being supportive of pedagogy. However, the visualisation might also be said to hide the data upon which it draws and the work done to represent it in that particular form. This might be argued to be enhanced, as educational environments are developed which are more digitally immersive and interactive. Here, we reach overlapping in research on education and digital gaming and simulation (e.g. Gros 2007).

Another example of the effects of code in curriculum-making is in the burgeoning field of learning analytics. Here, digital technologies mine data on students' past choices and behaviours to make predictions for the future, upon which recommendations can be made. At one level, this is held to have the potential to personalise learning and thereby enhance people's opportunities. At another, it might result in a person being 'trapped' by their past choices, when in education we might want to challenge and extend someone's opportunities. In relation to wider social and predictive analytics, the latter is referred to as a 'filter bubble' (Pariser 2011), where a person is argued to have their existing preferences reinforced through the work of code. Similar effects could arise from the use of learning analytics. However, more positively, there are attempts to develop analytics that, for instance, identify students, who more at risk of failing their programmes (e.g. http://analyse.kmi.open.ac.uk/project_info). Delegation, aggregation and quantification through the hidden work of code therefore have different possibilities and questions associated with them.

For code to work, there also needs to be available relevant digital databases, as 'the creation of digital archives are deeply computational in structure and content, because the computational logic is entangled with the digital representations of physical objects, texts and "born digital" artefacts' (Berry 2011, 25). Particularly significant, and yet at the same time largely unrecognised, is the role played by forms of classification and standardisation associated with the development of such databases, and the ways in which complex knowledge is represented (Lampland and Star 2009). This is not least because, as Manovich (2013, 215) points out, 'standardisation of file formats is an essential condition of interoperability between applications'. In other words, for the technology to function most flexibly, data and operations on data must be standardised. Classification requires developing standard forms of naming, as 'you can't store data without a classification system' (Bowker 2005, 140) and this requires naming and setting standards. Naming itself may seem a straightforward practice. Indeed, 'the activity of naming is mundane and low status, even though it is an activity central to the development of good databases' (Bowker 2005, 147). However, as Uprichard (2012) points out, the definitional purity of names and categories across time and space cannot be guaranteed; there is also the possibility of 'dirty data' in the practices of naming. In education, where much knowledge is contested, the dirtiness of data and the work to standardise might be said to result in a reduction of that contest; digital data is given an authority beyond that which is justifiable.

For Bowker (2005, 117), it is standards that 'undergird our potential for action in the world, both political and scientific; they make the infrastructure possible'. And without an infrastructure, action is not possible. Here,

> it is not just the bits and bytes that get hustled into standard form in order for the technical infrastructure to work. People's discursive and work practices get hustled into standard form as well. Working infrastructures standardize both people and machines. (Bowker 2005, 111–112)

But within this, 'each standard and each category valorizes some point of view and silences others' (Bowker and Star 2000, 5). Thus, it is in the entangled play of the visible and invisible that classification comes to order data, representations and practices. In education, how this occurs through the work of software and with what effects is largely left unexamined and unquestioned by those using the technology. However, if the work of software is opened, Bowker and Star (2000) suggest that this does not simply make things visible, but results in what they term a distribution of ambiguities. The question then is the extent to which all aspects of the visible and invisible in the standards are traceable or whether the ambiguities are possibly more inscrutable, enigmatic or mysterious. Tracing such ambiguities raises important methodological questions.

With the passing of time and the incorporation of digital data into new assemblages and applications, the pre-history of data; the selections and applications of standards; and the application of rules can disappear further from view. Data 'once encoded … can be resampled, transformed and filtered endlessly' (Berry 2011, 14). This can result in what Manovich (2013, 339, emphasis in original) refers to as data fusion, '*using data from different sources to create new knowledge* that is not explicitly contained in any of them … Combining separate media sources can also give additional meanings to each of the sources'. Representation and visualisation can become both richer and more hybrid. As the Web is developed, and with the advent of semantic technologies, which allow data to be shared, aggregated and reused across a linked web of databases and applications, any act of classification; any assumption encapsulated in a rule expressed in the code of a programme; or any decision to exclude certain results from the scope of a search may have implications far beyond its original setting. For example, if a relevant digital database is not open or readable within the terms of particular software, the analysis of a phenomenon and the data available may be incomplete and inaccurate. Or the data to be read may be questionable in quality or 'dirty'. For both curriculum and research practices, this raises important questions as to the extent to which we understand that which is enacted and upon what basis we evaluate its legitimacy.

Standards then, both those that allow software developers to reuse programme code in open source environments, and those that describe various kinds of 'content' contribute to an understanding of openness that privileges content reusability and technological interoperability. However, by insisting that things are described and knowledge represented in particular ways, potentially at the expense of a critical and exploratory pedagogical

openness, much would seem to be hidden from the teacher and learner, including the fact that the hiding has taken place. The picture here is therefore one of multiple complexities and is more than simply concern about the 'quality' or 'reliability' of information found online. The issue is not whether content from, for instance, *Wikipedia* or similar sources should find their way into educational settings, but of the multiple hidden translations, some effected by humans and some by software, that are incorporated into educational technology applications through the work of codes, algorithms and standards with what effects (Millerand and Bowker 2009).

All of this points to the increasing complexity of the work of software in curriculum-making practices, the tracing of which is and is likely to become ever more challenging. Not least, as, for instance, Manovich (2013, 198) argues in relation to digital representations,

> Unless you know how to program, you never encounter media content types – digital photos, digital videos, maps, etc. – by themselves. Instead, you encounter media content through particular software applications … 'representation' consists of two interlinked parts: data structured in particular ways and the interfaces/tools provided to navigate and work with this data.

Given this, it is perhaps little wonder that 'learning to code' has become an important rallying cry in and around education, in particular, schooling, in many contexts. The challenge for education has been put neatly as 'to programme or be programmed' (Naughton 2012). On this argument, as we saw above, in relation to the argument to support students to decipher the hidden curriculum more broadly, it is suggested that it is possible to make visible the work of software by developing and enhancing their computer skills. However, a question arises about the level at which one needs to learn to code to be able to meaningfully engage with its work and whether being able to do this also enables one to understand its multiple enactments and their significance. And the extent to which those who are good students or teachers in domains other than computing are able also to become good programmers.

'Learning to code' seems a somewhat simplistic response to a far deeper set of challenges, which may not be able to be addressed by individuals alone. What if making transparent the work of code, algorithms and standards is not entirely possible? What if, as Manovich (2013) suggests, reading the code is not as feasible as it is sometimes made out to be; that even the notion of 'computer literacy' is a misnomer? This would have profound implications for the practices and understanding of computer use in curriculum-making. Similar questions arise also from some recent research on computer algorithms. Barocas, Hood, and Ziewitz (2013) point out that algorithms have a history and geography of what they can and cannot do. They are neither stable, nor singular units of study or analysis. In line with

the wider social scientific research on software, they argue that 'algorithms are invoked as powerful entities that govern, judge, sort, regulate, classify, influence, or otherwise discipline the world' (Barocas, Hood, and Ziewitz 2013, 3). They also point to work that suggests that conventional disciplinary-based study, for example, computer science and sociology, places limitations on understanding the work of algorithms. In other words, to learn computer programming, as is currently suggested, would not necessarily result in an understanding of the full impact and significance of the work of software. From a research perspective, this also raises questions about both the theories and methods necessary to research this work (Kitchin and Dodge 2011).

Barocas, Hood, and Ziewitz (2013) suggest also that it becomes impossible to research the precise work of algorithms. They argue that the work of algorithms is very difficult to trace. It is inscrutable, elusive and mysterious. Here the work of software more broadly might be argued to be unknowable in any transparent sense. The performances of software become too complex and dynamic to be 'read' or fully understood. The work being done across space and time with different software and data-sets can be alluded to, but is itself elusive. If this is the case, then the impact of software as part of computer technologies may be more profound, as there is the possibility that they bring an inherent inscrutability into curriculum-making, with implications not least for ethical, political and legal responsibility. Here there is a sense in which aspects of the hidden curriculum associated with the work of software will always remain mysterious and, in some senses, hidden, the 'reading' of which may require more deconstructive strategies than more literal ways of reading for understanding. However, as I have suggested also, this hiddenness is not always necessarily a negative thing; black-boxing enables good and bad things to happen. From a literacy perspective, however, how do we know a closed book is a 'good' one?

Researching software in curriculum-making practices

'Turning everything into data, and using algorithms to analyse it changes what it means to know something' (Manovich 2013, 337). This stark assessment raises important issues for those students and teachers who are increasingly relying on digitalised data and interactions in their curriculum-making practices and researchers of digital education. As Manovich (2013, 338) goes on to argue,

> digital code, digital visualization, GIS, information retrieval, machine learning techniques, constantly increasing speed of processors and decreasing cost of storage, big data analytics technologies, social media, and other parts of the modern techno-social universe introduce new ways of acquiring knowledge, and in the processes redefine what knowledge is.

For Berry (2011, 4), the research challenge this raises is 'to bring software back into visibility so that we can pay attention to both what it is (ontology), where it has come from (through media archaeology and genealogy), but also what it is doing (through a form of mechanology)'. The latter is particularly important in relation to the work of education. There is a clear need to examine the ways in which software is entangled in curriculum-making practices in relation to forms of representation and the nature of knowledge, and the ways of interacting and knowing that are possible. The hidden curriculum of software may well be reproducing inequalities even as the ideology of digital education is increasingly emphasising openness.

However, this research agenda does depend upon an assumption that the work of software can indeed be made visible. The challenges are greater, if, as some are suggesting, the work of software becomes less traceable and more inscrutable. This is something Berry (2011, 5) himself acknowledges; 'looking at computer code is difficult due to its ephemeral nature, the high technical skills required of the researcher and the lack of analytical or methodological tools available'. However, he also argues that a phenomenology of computation can enable researchers to explore 'the ways in which code is able to structure experience in concrete ways' (Berry 2011, 39). This entails bringing the skills of the ethnographer together with those of the computer scientist, although even that is an oversimplification, as different manifestations of software require specific ethnographic and computer science methodologies, for example, those of particular types of literacies researcher.

The questions that emerge focus on the need to examine the multiple hidden and inscrutable translations that are entangled within curriculum-making practices through the enactments and uptakes of codes, algorithms and standards (Millerand and Bowker 2009), and the complex assemblings in enacting such work. This poses questions for educational researchers in exploring empirically the work of software in curriculum-making practices and whether there are processes at play that may not be simply hidden, but may be inscrutable, with multiple effects. Here Bowker (2005, 140) suggests that

> there has been little analysis of what happens as one gets from raw data to databases (and even less of the move from databases to analysis) – and whether decisions taken in this process have continuing effects on the interpretation and use of the resultant data stores.

This is particularly, significant if, as Bowker (2005, 152) also suggests, the world with which one engages 'becomes more and more closely tied to the world that can be represented by one's theories in in one's databases; and this world is ever more readily recognised as the real world'. In other words, the world that is simulated through software becomes the real world

to those engaged in educational practices. The world is made real through the simulations that are enacted; what Latour (2010) refers to as the production of 'factishes'. On this reading, all digital curriculum-making become forms of simulation. The increasing research on the code, standards and algorithms in daily life and work raise important questions for research on digital education and curriculum-making practices that need further exploration empirically and conceptually. Methodologically, this raises some major challenges that require an understanding of software as well as education. Pedagogically, this requires increased engagement with the active role of software in the enactments of curriculum and more explicit discussion of the legitimacy and illegitimacy of different forms of the hidden curriculum in the practices and relations of digitalised curriculum-making. This article is intended as an opening into this area of work.

Acknowledgements

My thanks to the referees of this article who provided valuable suggestions for its development. This article is developed from work within the ESRC-funded seminar series Code Acts in Education http://codeactsineducation.wordpress.com/about/ (grant reference: ES/L001160/1).

References

Anderson, T. 2001. "The Hidden Curriculum in Distance Education: An Updated View." *Change* 33 (6): 29–35.

Apple, M., and N. King. 1983. "What Do Schools Teach?" In *The Hidden Curriculum and Moral Education*, edited by H. Giroux and D. Purpel, 143–167. Berkeley, CA: McCutchan.

Barocas, S., S. Hood, and M. Ziewitz. 2013. "Governing Algorithms: A Provocation Piece." Paper prepared for the Governing Algorithms Conference, New York University, New York, May 16–17.

Berry, D. 2011. *The Philosophy of Software: Code and Mediation in the Digital Age*. Basingstoke: Palgrave Macmillan.

Bourdieu, P., and J.-C. Passeron. 2000. *Reproduction in Education, Society and Culture*. London: Sage.

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

Bowker, G., and S. Star. 2000. *Sorting Things Out: Classification and Its Consequences*. Cambridge, MA: The MIT Press.

Cope, B., and M. Kalantzis, eds. 2000. *Multiliteracies*. London: Routledge.

Cotton, D., J. Winter, and I. Bailey. 2013. "Researching the Hidden Curriculum: Intentional and Unintended Messages." *Journal of Geography in Higher Education* 37 (2): 192–203.

Edwards, R., and P. Carmichael. 2012. "Secret Codes: The Hidden Curriculum of the Semantic Web." *Discourse* 33 (4): 575–590.

Eriksson-Zetterquist, U., K. Lindberg, and A. Styhre. 2009. "When the Good times Are over: Professionals Encountering New Technology." *Human Relations* 62 (8): 1145–1170.

Farrell, L. 2006. *Making Knowledge Common: Literacy and Knowledge at Work*. New York: Peter Lang.
Faulkner, P., C. Lawson, and J. Runde. 2010. "Theorising Technology." *Cambridge Journal of Economics* 34: 1–16.
Ferguson, R., and S. Buckingham Shum. 2012. "Social Learning Analytics: Five Approaches." Proceedings of the 2nd International Conference on Learning Analytics & Knowledge, Vancouver, BC, April 29–May 2.
Gros, B. 2007. "Digital Games in Education: The Design of Games-based Learning Environments." *Journal of Research on Technology in Education* 40 (1): 23–38.
Kitchin, R., and M. Dodge. 2011. *Code/Space: Software and Everyday Life*. Cambridge, MA: The MIT Press.
Lampland, M., and S. Star, eds. 2009. *Standards and Their Stories: How Quantifying, Classifying, and Formalizing Practices Shape Everyday Life*. Ithaca, NY: Cornell University Press.
Latour, B. 2010. *On the Modern Cult of the Factish Gods*. Durham: Duke University Press.
Law, J., and W. Bijker. 1992. "Postscript: Technology, Stability and Social Theory." In *Shaping Technology/Building Society*, edited by W. Bijker and J. Law, 290–308. Cambridge, MA: The MIT Press.
Lawn, M., ed. 2013. *The Rise of Data in Education Systems*. London: Symposium Books.
Lawn, M., and I. Grosvenor, eds. 2005. *Materialities of Schooling*. Oxford: Symposium Books.
Loveless, A., and B. Williamson. 2013. *Learning Identities in a Digital Age: Rethinking Creativity, Education and Technology*. London: Routledge.
Manovich, L. 2013. *Software Takes Command: Extending the Language of New Media*. London: Bloomsbury Academic.
Margolis, E., ed. 2001. *The Hidden Curriculum in Higher Education*. London: Routledge.
Millerand, F., and G. Bowker. 2009. "Metadata Standards: Trajectories and Enactment in the Life of an Ontology." In *Standards and Their Stories: How Quantifying, Classifying, and Normalizing Practices Shape Everyday Life*, edited by M. Lampland and S. Star, 149–176. Ithaca, NY: Cornell University Press.
Naughton, J. 2012. "A Manifesto for Teaching Computer Science in the 21st Century." *The Observer*, March 31. http://www.guardian.co.uk/education/2012/mar/31/manifesto-teaching-ict-education-minister.
Nespor, J. 2006. *Technology and the Politics of Instruction*. London: Routledge.
Orlikowski, W. 2007. "Sociomaterial Practices: Exploring Technology at Work." *Organization Studies* 28 (9): 1435–1448.
Orlikowski, W., and C. Iacono. 2001. "Research Commentary: Desperately Seeking the 'IT' in IT Research – A Call to Theorizing the IT Artifact." *Information Systems Research* 12 (2): 121–134.
Pargman, D., and J. Palme. 2009. "ASCII Imperialism." In *Standards and Their Stories: How Quantifying, Classifying, and Formalizing Practices Shape Everyday Life*, edited by M. Lampland and S. Star, 177–199. Ithaca, NY: Cornell University Press.
Pariser, E. 2011. *The Filter Bubble: What the Internet is Hiding from You*. London: Penguin Books.
Snyder, B. 1971. *The Hidden Curriculum*. New York: Knopf.
Snyder, I., ed. 2002. *Silicon Literacies*. London: Routledge.

Star, S., and M. Lampland. 2009. "Reckoning with Standards." In *Standards and Their Stories: How Quantifying, Classifying, and Formalizing Practices Shape Everyday Life*, edited by M. Lampland and S. Star, 3–24. Ithaca, NY: Cornell University Press.

Thrift, N. 2004. "Remembering the Technological Unconscious by Foregrounding Knowledges of Position." *Environment and Planning D: Society and Space* 22 (1): 175–190.

Thrift, N. 2005. "Beyond Mediation: Three New Material Registers and Their Consequences." In *Materiality*, edited by D. Miller, 231–255. Durham, NC: Duke University Press.

Uprichard, E. 2012. "Dirty Data: Longitudinal Classification Systems." *Sociological Review* 59: 93–112.

Woolgar, S. 1991. "Configuring the User: The Case of Usability Trials." In *A Sociology of Monsters: Essays on Power, Technology and Domination*, edited by J. Law, 57–99. London: Routledge.