While technically possible to be cocooned in everyware, it is also sometimes difficult to be connected even in developed economies. A recent 3-day train trip across the United States showed how variable mobile phone connections can be with nonexistent Wi-Fi access along the route. Thankfully, the author found occasional unsecured systems along the way but, in general, the experience showed the absence of everyware as well as the withdrawal suffered by the usually connected being disconnected for 3 days. The authors' acknowledge the many spatial gaps in everyware and, in this chapter, further identify some of the threats as well as benefits of ubiquitous computing.

Befitting its status as one of the first syntheses of the spatial context of software, the final chapter addresses how future study of this phenomenon might evolve. As with so many fields of inquiry, analysis requires the collaboration of many disciplines—engineering, science, social science, and humanities—to fully identify and understand the meaning of code/space. Kitchin and Dodge recognize the value in an expanded mindset to see code/space as a complex and sociocultural product.

One important element in the realm of code/space is the personal experience individuals have with software, not only in terms of life impacts but also in its application. Early PCs required users to have some familiarity with the structure of the software they used and to determine how and when it was changed or upgraded. Today, so much of the relationship between user and software is distanced by the complexity and design of software systems. The complexity is underscored by the sobering announcement of one's laptop that it has 16,142 upgrades to apply while explaining to airline personnel that Microsoft has ordered me not to turn off my computer. It is usually at this juncture that Adobe and Java also announce their need to upgrade my system. The end result is the distancing of the user from knowing how and what happens as remote software systems take control. While I am relieved, I do not need to manually manage 16,142 upgrades to my laptop, I also feel disconnected from what is happening, and whether or not the next iteration will be an improvement in my eyes.

Code/Space is engaging and thought provoking and offers language and concepts for geographers to examine a new spatial dimension. It also makes visible a significant, yet invisible, force in the shaping of daily life. If I were to cavil, it would be that the text seems more positive about software than many critics would allow, although in its defense Kitchin and Dodge do acknowledge some of the problems associated in our surrender to code. Important elements of code/space deserving similar treatment would be medicine and military uses of software. *Code/Space* is a welcome addition to the scholarship on cybergeography and a valuable outline for the future spatial analysis and understanding of software.

Reference

Morone J (1998) *The Democratic Wish: Popular Participation and the Limits of American Government*. New Haven, CT: Yale University Press.

Code/space and the nature, production and enrolment of software

Reviewed by: Martin Dodge, *University of Manchester, UK, and Rob Kitchin, National Institute for Regional and Spatial Analysis, National University of Ireland, Maynooth, Ireland*

In the four or so years since we completed the manuscript of *Code/Space*, it is evermore apparent that the onrushing pace of digitisation of economic transactions and social interactions means that scope of software has increased in most people's lives, yet the ability to comprehend the workings of code itself and account for its agency remains limited. Given the growing automation of activities, the potential

advantages at least for some, and the inherent risks of taking humans out of the loop, it is helpful to have this forum to consider why code matters from a spatial perspective and how human geographers might contribute to documenting work of software.

Before we respond to their critiques, we would like to thank Paul Adams, Aaron Kellerman, Sam Kinsey and Mark Wilson for reading and ruminating on our book *Code/Space*. Collectively, they provide useful reflections on our arguments and additional empirical scope that challenge us to extend our thinking and analysis in productive ways. In this short response, we extract and synthesise what we think are their most valuable insights and in the final section detail how we are about to embark on a new project – The Programmable City – that seeks to perform some of the work they call for.

From our reading, the reviews suggest three issues that *Code/Space* neglected or failed to fully explicate: the nature, consumption and production of software.

In *Code/Space*, we developed an argument that software is both a product and producer of the world, a special kind of technology that possesses secondary agency that enables and empowers at the same time as it introduces a new form of governance, automated management, wherein software enacts automatic, automated and autonomous actions. Whilst sympathetic to this approach, the reviewers correctly suggest that further work is required in thinking through the nature of software. Kellerman argues that code constitutes 'a special kind of information' and needs to be 'assessed against other forms of information and their geographies.' It seems to us that the executable nature of machine code running on digital computers is distinctive from other forms of information, although the case could be made that in essence, it is not too different to a punched paper tape directing a mechanical weaving machine. Given the long genealogy of such self-instructional code, we did not explore code within a framework of information theory, but such framings would undoubtedly be useful and productive (e.g. Cox, 2013). Kinsey suggests a deeper engagement with Bernard Stiegler's work, theories of technogenesis, philosophies of technology more broadly, and the role of the speculative in the discursive formation of software, all of which we concur with. Adams rightly makes the case for a greater focus on the nature of software's secondary agency and the role of programmers in producing code, which inevitably inculcates their values, ideologies and desires.

The latter leads into what Kellerman notes is a second underdeveloped theme, that of the production of software. He notes that in *Code/Space*, we neglect an analysis of the geographies and political economy of the research and development, creation and distribution of software products. We did note that code not only reflects the secondary agency of programmers, but is a complex techno-social product bound up in diverse markets, capitalist modes of accumulation, constraints of regulations and laws and discourses of openness and sharing in the case of open-source software, but in *Code/Space*, we focused our attention primarily on the work that software does in the world, rather than providing an examination of how code is made. We agree with Adams and Kellerman's assertions that much more research needs to be conducted into the creation of code, both in relation to the investment of secondary agency, but also its wider contextualisation as localised and globalised product. A focus on the latter and the geography of the software industry has been and is being actively researched in business schools, economic geography and regional studies, with linkages to more applied planning concerns for attracting information technology (IT) jobs and stimulating so-called 'creative industries'.

The third underdeveloped theme noted in the reviews concerns the consumption of software and how software inflects individual agency and sense of self. Kellerman posits that it would be profitable to examine the geographies of software consumption across a number of socio-spatial contexts such as level of development and political regime. He rightly argues that we presently know little about how software consumed in different locales and what factors shapes its consumption. Wilson reflects on the extent to which people interact actively with software, writing their own code or configuring settings and how this is changing over time and circumstance and what it means for the work code does. Certainly, there is much more to be revealed about the individual psychology of software

enrolment and the significance of spatial context in situating interactions, especially if/when wearable computers (e.g. Google Glasses) become the cultural norm. Similarly, Adams points out that we did not pay enough attention to 'our status and nature as code-users' and 'the kind of agents we become by inhabiting these [code] spaces'. He suggests it would be productive to examine the intersection between the primary agency of software users, and those that software seeks to act upon, and the secondary agency of software and its developers, and to think through 'how code permits people to act at a distance and over time, more often, with more ongoing feedback, and in more complex ways, than they were able to do without code.' For Kinsey, the larger philosophical question that needs answering is 'what it means to be human in an age of a perceived increase in technological agency'? We agree that much more research and thinking needs to be conducted with respect to the emergent relationship between code and people and how each shapes the other in relational and contingent ways.

We were aware of some of these gaps in the book and made brief reference to them in the concluding chapter. Subsequently, we have outlined a detailed programme of research that seeks to address the issues raised by the reviewers as well as extend our original work (Kitchin, 2011). This programme suggests using the city as a platform for investigating processes of software translation (how cities are translated into code) and transduction (how code reshapes city life) (see Figure 1), operationalised through eight questions concerning how software inflects how we know, manage, work and live in cities (see Table 1). Formulated in this gridded fashion, the questions work both vertically and horizontally to frame a programme of research that should provide further insights into the nature, production and consumption of software.

This programme of research has recently received 5 years of funding from the European Research Council and will employ a team of four postdoctoral researchers and four doctoral students, each focused on one of the broad analytic questions. Our hope is that this set of interlinked projects will produce a large body of empirical material as well as providing new theoretical insights that illuminate the

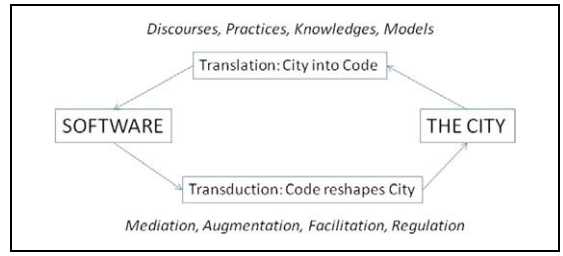

Figure 1. The conceptualisation of the 'Programmable City' (Kitchin, 2011).

nature of software, how it is produced and consumed and the work it does in the world.

Yet, getting a clearer map of code/space is going to be challenge for several reasons, not least because of the speed of development of software products and services, and the continued invisibility of the executable code and data structures beneath the surface of the interface. Moreover, like much social science research, there will be issues of access, with key aspects of scholarly analysis being controlled by institutions, companies and third parties that have little need to cooperate and may actively resist outside scrutiny of their commercial secrets or vital operations. The temptation will be to analyse more accessible forms of code (such as online media and open source software development) rather than investigating the code that matters most to daily life and the ongoing production of the city (e.g. control systems of the utility companies, the scheduling software of transport systems, the tasking of security personnel, the calculation of insurance rates and mortgage risks, etc.). A critical challenge is the need to expose algorithmic pinch-points and demonstrate in precise ways how a piece of code makes space come into being. Even if such research becomes possible, a further problem is how to communicate its significance to a wider audience without resorting to logical operations and mathematical expressions. (An effort in this direction, Cormen, 2013, is worthwhile but demonstrates how hard it is to do in practice.) This is the 'elephant in the room' for software studies given that most social science and humanities scholars cannot programme, raising an epistemological problem in researching code. Without the ability to both make and deprogramme code, can social scientists really begin to show the

Table 1. An analytical framework for a comprehensive account of the work of software in contemporary Western urban contexts.

	Translation: City into code	Transduction: Code reshapes city
Understanding the city (Knowledge)	How are digital data generated and processed about cities and their citizens?	How does software drive public policy development and implementation?
Managing the city (Governance)	How are discourses and practices of city governance translated into code?	How is software used to regulate and govern city life?
Working in the city (Production)	How is the geography and political economy of software production organised?	How does software alter the form and nature of work?
Living in the city (Social Politics)	How is software discursively produced and legitimated by vested interests?	How does software transform the spatiality and spatial behaviour of individuals?

consequences of code for everyday activities? Such abilities might be vital if we are to have a meaningful influence on the development of code and coding practices and to resist software that is unethical and open channels for the progressive production and use of software. If this is the case, then perhaps we are failing to equip the next generation of researchers with the skills required for making sense of a world inflected with software; whilst many students are Web savvy and adroit in using software, they are largely ignorant of how to produce it. Perhaps then the real follow-on from *Code/Space* is need for someone to write a book not on the geographies of code but a textbook to teach geographers to code.

References

Cox G (2013) *Speaking Code*. Cambridge, MA: MIT Press.

Cormen TH (2013) *Algorithms Unlocked*. Cambridge, MA: MIT Press.

Kitchin R (2011) The programmable city. *Environment and Planning B* 38: 945–995.