Environment and Planning B: Planning and Design 2011, volume 38, pages 945 – 951

doi:10.1068/b3806com

Commentary

The programmable city

"[T]he city itself is turning into a constellation of computers."
Batty (1995)

"The modern city exists as a haze of software instructions. Nearly every urban practice is becoming mediated by code."
Amin and Thrift (2002, page 125)

Software is essential to the functioning of cities. It is deeply and pervasively embedded into the systems and infrastructure of the built environment and in the management and governance of urban societies. Software mediates how we understand and plan cities, how we manage urban services and utilities, and how we live urban lives. As a result, across a diverse set of everyday tasks—domestic chores, work, shopping, travelling, communicating, governing, and policing—software makes a difference to how social, spatial, and economic life takes place. Such is software's capacities and growing pervasiveness that some analysts predict that we are entering a new phase of 'everyware' (Greenfield, 2006); that is, computational power will be distributed and available at any point on the planet.

The phenomenal growth in software creation and use is due to its emergent and executable properties: how it codifies the world into rules, routines, algorithms, and databases, and then uses these to do work in the world to render aspects of everyday life programmable. Whilst it is not fully sentient and conscious, software can exhibit some of the characteristics of 'being alive' (Thrift and French, 2002). This property is significant because code enables technologies to do work in the world in an autonomous fashion—that is, it can process data, evaluate situations, and make decisions without human oversight or authorisation. In other words, it possesses what Mackenzie (2006) terms 'secondary agency'. However, because software is embedded into objects and systems in often subtle and invisible ways, it largely forms a 'technological unconscious' that is only noticed when it performs incorrectly or fails (Thrift, 2004). As a consequence, software often appears to be 'automagical' in nature, in that it works in ways that are not clear and visible, and it produces complex outcomes that are not easily accounted for by people's everyday experiences (Kitchin and Dodge, 2011).

And yet, despite its importance, software, the assemblages that support and promote its production and adoption, and the work that it does in the world are barely theorised and empirically studied from a social sciences perspective. Instead, software has been understood from a technical, instrumental perspective that treats it as largely an immaterial, stable, neutral product, rather than as a complex, multifaceted, mutable set of relations created through diverse sets of discursive, economic, and material practices. Where the role of software has been acknowledged, the focus of analysis has been the technologies and infrastructures that software enables, rather than the underlying nature of software that powers such technologies. The consequence is to study how telematic networks shape, for example, traffic management, but to largely ignore how such effects are manifestly the result of the rules and procedures formalised within the code of management software. The way the discourses and practices of traffic management are translated into the routines and algorithms of code is vitally

important to how the traffic system operates and yet we know hardly anything about how such translations occur: traffic system into code; code reshaping the traffic system.

Indeed, we know very little about the ways in which software is socially created; the nature of software itself; how discourse, practices, and knowledge get translated into algorithms and code; the geographies and political economy of software development; how software is embedded into various social systems; how software applications work with each other to create complex assemblages that work within and across scales; the power wielded through software's 'secondary agency'; and how software alternatively modulates the production of space and transforms the nature of governance. Nor do we have a good understanding of the social implications of a number of different forms of computing (pervasive, ubiquitous, sentient, tangible, wearable) that are presently being developed and rolled out.

My argument in this commentary is about the need for a sustained programme of research on the nature of software and contemporary urbanism, and in particular an analysis of the two core interrelated aspects of the emerging programmable city: (a) translation: how cities are translated into code; and (b) transduction: how code reshapes city life (see figure 1). One productive way to organise these analyses might be to frame them with respect to key urban practices. Such an initial analytic framework is outlined in table 1, focusing on understanding, managing, working, and living in the city, which is elaborated in the rest of the commentary.

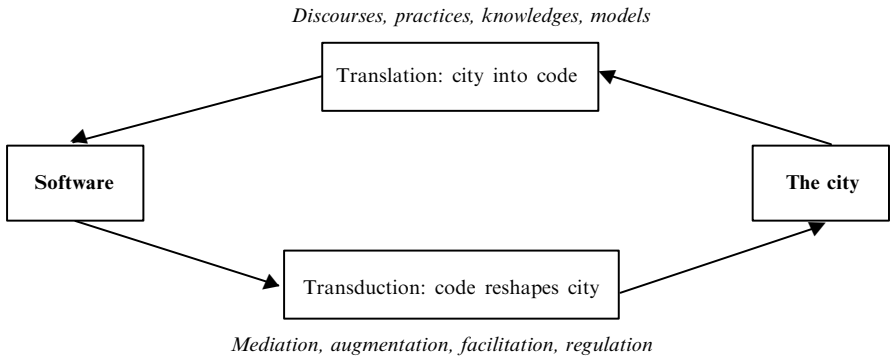

Figure 1. The programmable city.

Table 1. A proposed analytical framework for investigating the programmable city.

	Translation: city into code	Transduction: code reshapes city
Understanding the city	How are digital data generated and processed about cities and their citizens?	How does software drive public policy development and implementation?
Managing the city	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	How does the geography and political economy of software production shape coding practices?	How does software alter the nature of work?
Living in the city	How is software discursively produced and legitimated by vested interests?	How does software transform the spatiality of sites?

Whilst an examination of the relationship between software and cities needs to build on theoretical ideas and established literatures from geography, urban studies, and sociology it is critical that the analysis of programmable urbanism draws strongly from the perspective of the fledgling field of software studies (Berry, 2011; Fuller, 2008; Galloway, 2004; Kitchin and Dodge, 2011; Mackenzie, 2006; Manovich, 2000; 2008). In broad terms, this means focusing on software per se—its nature, composition, coding practices, political economy, spatialities—and not simply the technologies it enables. This shift in focus is equivalent to the difference between studying the underlying epidemiology of ill health and the effects of ill health on the world. Whilst one can understand the relationship between health and society by studying how ill health affects social relations, one can gain deeper insights by considering the specifics of different diseases, their aetiology, and how these manifest themselves in shaping social relations. Software studies then focuses on the aetiology of code, and how code makes digital technologies what they are, and shapes what they do. It seeks to open the 'black box' of processors and algorithms to understand how software—its lines and routines of code—does work in the world by instructing technologies and people how to act. More specifically, it means conceptualising software as both a product of people in time and space and a transducer of time and space.

The city into code

How are digital data generated and processed about cities and their citizens?

Societies have always generated information about their citizens and environments for the purposes of monitoring and managing them. With the advent of software-enabled technologies, more and more data about individuals, their lives and lifestyles, city infrastructure and services, and the built environment are being generated, stored, and analysed in new ways. First, there have been significant advances in the 'tagging' of people, objects, information, transactions, and territories to make them more amenable to capture by making them machine readable and to increase the granularity of capture. Second, the technologies to record data have been widened to include a range of scanning and sensing devices with associated increases in accuracy, sophistication, distribution, and form. The recording of data is thus more effective and efficient, often recorded on a continuous basis, with measurement often mobile, dynamic, in real time, networked, and distributed (Dodge and Kitchin 2005a). Third, there have been significant advances in the virtual and physical storage of data, along with a decline in size and cost overheads. Whilst there has been a substantial amount of work examining the production and analysis of data in the digital age, there is still the need for a comprehensive, critical, cross-sectoral, and spatial analysis of how cities and their citizens are ontologically framed, categorised, and captured. Such an analysis is important because such data provide the raw materials that code works upon, and thus structure the kinds of analysis and actions that can take place.

How are discourses and practices of city governance translated into code?

Code is the manifestation of a system of thought—an expression of how the world can be captured, represented, processed, and modelled computationally. Programming captures and enacts knowledge about the world—practices, ideas, rules, measurements, locations, equations, images—abstracting the world into defined, stable ontologies of data and sequences of commands that define relations between data, and details how those data should be processed. The production of such software is not simply a technical exercise, as framed by much of computer science. Rather it is a complex and contingent process, shaped by the abilities and worldviews of programmers and system designers working in companies situated in social, political, and economic contexts (Rosenburg, 2007). At present, we have little understanding of how the

discourses, legalities, and practices of urban life and regulation are translated into software both with respect to how software developers work to effect such a translation and how data about the world are conceptualised within the framework of programming beyond a software engineering approach which focuses solely on the technical aspects of such a translation. There is a critical need, then, for detailed studies of how developers produce code and the life of software projects in order that we can build a better understanding of the ways in which software is diversely created.

How does the geography and political economy of software production shape coding practices?

Software development, like any industry, has a particular space economy to its production (Boschma and Weterings, 2005). Its operations are organised and structured both sectorally and spatially to enable efficient production and minimise costs and maximise profit. Whilst software creation takes place across the world, it is concentrated in key sites (for example, Silicon Valley, Bangalore, and Ireland), where there is an agglomeration of skilled labour, entrepreneurship, technologies, venture capital, and tax and development regimes. At these sites, software is the product of a complex interplay of social, political, and economic processes operating within and across scales. For other kinds of software production, such as open-source projects, development is often more dispersed and distributed. Indeed, as Mackenzie (2003, page 3) notes, software is produced through "complex interactions involving the commodity production, organizational life, technoscientific knowledges and enterprises, the organization of work, manifold identities and geo-political-technological zones of contact." The creation of software is thus placed both with respect to the local milieu and also scaled into regional and global networks. To date, there has been little research examining how the creation of software is shaped and scaled by economic and spatial processes. Rather, research has been directed from a business and computing perspective, often focusing on organisational matters but lacking a wider contextual framing.

How is software discursively produced and legitimated by vested interests?

The adoption of software and digital technologies, and the systems they underpin, has been complemented by a broad set of discourses and political mobilisation that have sought to justify their development and naturalise their use. With respect to the rollout of software solutions, typical rationales include that software will increase efficiency, productivity, reliability, flexibility, competitive advantage, profitability, security, and safety, and will tackle crime and fraud (Dodge and Kitchin, 2005a). Remarkably little work has examined the ways in which these discourses are bound together in discursive regimes by a variety of vested interests—government, corporate, and civil—in order to propagate the adoption of software-enabled technologies. There is a need, then, to explore how discursive regimes are built through alliances; how they are advanced through different media such as advertising, press coverage, online media, trade fairs, staff training, and lobbying; how discursive regimes are supported by legislation and quasi-legal conventions that legitimate the governmentality that code enacts; how the discourses promoted are countered by those who question their logic and social implications; and how the interchange between these sides unfolds to shape how software is developed, deployed, and received.

Code reshapes the city

How does software drive city policy development and planning?

It is now standard practice that urban information systems and spatial data infrastructures are used to organise, process, and present data concerning cities and their citizens in order to provide an evidence and analysis base for public policy development,

implementation, and monitoring. Such systems form a core part of city management and foresight exercises related to all aspects of socioeconomic development through to localised planning and urban design. Indeed, it is now possible to collate and cross-reference huge quantities of data, to synthesise and analyse such data, model and simulate potential outcomes, and to output findings in a variety of formats from tables to charts to maps. As a result, information systems enable more sophisticated and timely urban analysis, to ask new questions, and to answer those questions in new ways. Whilst there has been work within fields such as critical GIS on the power and role of new information systems in shaping public debate and the policy process (eg, Pickles, 1995), much of this work remains at a fairly abstract and political level, rather than charting in detail, using empirical studies, how such systems are mobilised as key discursive resources in shaping policy formulation and implementation.

How is software used to regulate and govern city life?

Over the past two centuries a mode of governmentality has developed in Western society that is heavily reliant on generating and monitoring systematic information about individuals by institutions. Software-enabled technologies qualitatively alter both the depth and the scope of this disciplinary gaze, but also introduce new forms of governance, because they make the systems and apparatus of governance more panoptical in nature. At the technical level, software is producing new machine-readable and software-sorted geographies that are radically altering how cities are regulated (Dodge and Kitchin, 2005a, Graham, 2005). Software creates more effective systems of surveillance and creates new capture systems that actively reshape behaviour by altering the nature of a task. In recent years there has been much academic attention paid to qualitative changes in surveillance technologies as they have become digital in nature, leading to the development of a new field of surveillance studies. That said, there is still much conceptual and empirical work to be done to understand how forms of governance are being transformed and the role played by software, and not simply the broader technologies they enable. What are desirable are in-depth case studies that examine how software is deployed to regulate and govern city life; how managers of city services understand and use the software at their disposal; how and why people adopt and submit to forms of management; plots fully the complex and contingent ways that people understand and react to the discursive and affective fields of software; and how people use and resist forms of software-mediated management.

How does software alter the nature of work?

Software has clearly had an impact on how work is performed, workplace culture, and the structural and spatial organisation of workplaces and companies. Software mediates all kinds of work tasks across primary, manufacturing, and service sectors, particularly through the use of software-enabled, information, communication, and management technologies. Many workers, including academics, are software workers, using software packages to capture and process data and produce analysis and transmit their work. Software is enabling new work practices, including work on the move. To a significant degree, code is the structural 'glue' that binds distributed and distanciated activities such as logistical chains together. Whilst there has been a vast amount of research undertaken on how organisations are adopting and utilising software-enabled technologies, principally from a business and management perspective, it has little considered the ways in which the software actually functions and sociospatially structures work practices and processes. There is a need to address this lacuna by examining in detail how the work at different sites (such as offices, shops, and hospitals) is being transduced by code; how the sociospatial practices and organisation of work

in these locales are being reconfigured and rescaled by software; and how software interfaces, algorithms, and data capture and processing alter the tasks, forms, spaces, and scales of work.

How does software transform the spatiality of sites?
The nascent work of software studies to date has focused on the role of software in social formation, organisation, and regulation, as if people and things exist in time only, with space a mere neutral backdrop. What this produces is historically nuanced, but largely aspatial accounts of the relationship between software and society. Urban life, however, does not operate independently of space. Software and the work it does are the products of people and things in time and space, and it has consequences for people and things in time and space. To date, there have only been a handful of studies that have examined how software transforms spatiality (eg, Budd and Adey, 2009; Crang and Graham, 2007; Dodge and Kitchin, 2005b; Graham, 2005; Kitchin and Dodge, 2011; Thrift and French, 2002). There is, therefore, a paucity of rich empirical material and there are numerous outstanding questions concerning the relationship between software and space with respects to different sites—the home, the farm, the factory, the station, the public plaza, the street, and so on.

Conclusion

In the years since Batty (1995) noted that the city was "turning into a constellation of computers" there has been much analysis considering the effects of information and communication technologies on cities and their form and functions. To date, however, these analyses have focused largely on the effects of software-enabled technologies, but have little considered software itself in mediating such effects. My argument has been that such a consideration of the nature, composition, political economy, and spatialities of software—an unpacking of its aetiology—is important in order to more fully understand how the world, and its practices, ideas, rules, measurements, locations, equations, images, and so on, are captured and worked on by software, and how such software does work in the world in diverse ways by instructing various technologies and people how to act. In other words, I have made the case that we are entering a period of programmable urbanism, and that to understand this new form of urbanism we need to examine the various components of how the city is translated into code and how the resulting software is reshaping city life—that we need to understand the internal workings of the black box in order to better understand its external work. In so doing we can start to address a series of important lacunae in understanding and theorising contemporary urbanism, opening up new comprehensions of the city at a time when urban life is going through profound changes with respect to its organisation, scaling, and management.

Rob Kitchin
NIRSA, National University of Ireland Maynooth, County Kildare

References

Amin A, Thrift N, 2002 *Cities: Reimagining the Urban* (Polity, London)
Batty M, 1995, "The computable city" *International Planning Studies* **2** 155 – 173
Berry D M, 2011 *The Philosophy of Software: Code and Mediation in the Digital Age* (Palgrave Macmillan, London)
Boschma R, Weterings A, 2005, "The effect of regional differences on the performance of software firms in the Netherlands" *Journal of Economic Geography* **5** 567 – 588
Budd L, Adey P, 2009, "The software-simulated airworld: anticipatory code and affective aeromobilities" *Environment and Planning A* **41** 1366 –1385
Crang M, Graham S, 2007, "Sentient cities: ambient intelligence and the politics of urban space" *Information, Communication and Society* **10** 789 – 817

Dodge M, Kitchin R, 2005a, "Codes of life: identification codes and the machine-readable world" *Environment and Planning D: Society and Space* **23** 851 – 881
Dodge M, Kitchin R, 2005b, "Code and the transduction of space" *Annals of the Association of American Geographers* **95** 162 – 180
Fuller M, 2008 *Software Studies: A Lexicon* (MIT Press, Cambridge, MA)
Galloway A R, 2004, *Protocol: How Control Exists After Decentralization* (MIT Press, Cambridge, MA)
Graham S, 2005, "Software-sorted geographies" *Progress in Human Geography* **29** 562 – 580
Greenfield A, 2006 *Everyware: The Dawning Age of Ubiquitous Computing* (New Riders, Boston, MA)
Kitchin R, Dodge M, 2011 *Code/Space: Software and Everyday Life* (MIT Press, Cambridge, MA)
Mackenzie A, 2003, "Transduction: invention, innovation and collective life", www.lancs.ac.uk/staff/mackenza/papers/transduction.pdf
Mackenzie A, 2006 *Cutting Code: Software and Sociality* (Peter Lang, New York)
Manovich L, 2000 *The Language of New Media* (MIT Press, Cambridge, MA)
Manovich L, 2008 *Software Takes Command*, http://lab.softwarestudies.com/2008/11/softbook.html
Pickles J, 1995 *Ground Truth: The Social Implications of Geographic Information Systems* (Guilford Press, New York)
Rosenberg S, 2007 *Dreaming in Code: Two Dozen Programmers, Three Years, 4,732 Bugs, and One Quest for Transcendent Software* (Three Rivers Press, New York)
Thrift N, 2004, "Remembering the technological unconscious by foregrounding knowledges of position" *Environment and Planning D: Society and Space* **22** 175 – 190
Thrift N, French S, 2002, "The automatic production of space" *Transactions of the Institute of British Geographers, New Series* **27** 309 – 335