Software Writing Cities

Address given by Nigel Thrift

Professor of Geography, University of Bristol

"Information and the Urban Future"

Taub Urban Research Center, New York University

February 26, 2001

Introduction

Perhaps I'd better start by explaining why I became interested, first of all, in geography. One of the reasons is theoretical, and stems from the work of Bruno LaTours - he wrote a very nice book called "Paris, Invisible City", in which he was trying to show the way in which all the mundane objects in everyday life - street signs, the double yellow line - all of these different kinds of things directed human bodies to go in a particular way without them really having to think about just what they did. And it's that kind of insight that I want to look at in terms of software.

The second reason is practical, and that is University of Bristol [U.K.] where I come from, has just been awarded a very large amount of money by the U.K. government to set up a center for communications, computing and content, which is based on Bristol's wireless systems. I've had a bit to do with this and the result is very interesting indeed for wireless technology. And so, in a sense, this paper is a kind of confluence of those two components.

What I want to argue is that what we're starting to see coming into being is an age of what one might call mechanical writing. That pertains to software, which has a kind of a passion for inscription. My colleague, Shawn French, and I believe that the gradual evolution of software is extraordinarily important to understanding the current direction of Euro-American cities, and especially the way in which these cities are increasingly being beckoned into existence by code itself. One of the problems is to try and understand what that might actually mean. And one of the problems and one of the strangenesses in this literature, is how remarkably unremarked this massive change actually is.

There are four reasons why this degree of neglect has happened. One of them is the fact that software takes up very little in the way of physical space. It generally occupies micro-spaces, and isn't noticeable on that level. Secondly, it is deferred. It expresses the kind of co-presence of different kinds of times: the time of its production, and the subsequent dictation of future moments. I'm going to talk especially about that, because I think that this, in a sense, is where the politics of software comes in. Thirdly, software is constantly in a state of 'in-between'. It's a whole series of instructions that lie in the interstices of everyday life. Fourthly, of course,

we are schooled to ignore software, just as we are schooled to ignore lots of different kinds of everyday objects.

Now, one could get fancy at this point if one wanted, and call on all kinds of notions of ego and this kind of thing. We don't want to do that, though; what we want to do instead, is to talk about software, as a kind of absorption, and I think one of the reasons why this absorption is so little-noticed, is that we still think in terms of a city populated by humans and by objects which represent each other via words and images. And that makes it difficult, I think, to mark out software's territory, and yet incredibly important to try and do so. And what we want to see software as, therefore, is not so much a text as in fact a whole series of presentations, a series of 'writing acts' of one form or another. We're going to call on two key theoretical technologies to try to understand this.

One of these is the kind of work that revolves around network theory, and the other is the kind of work which is trying to understand writing as performances in one way or another. And if we marry these two kinds of theories together, we might start to be able to get a closer idea of what it is that software can tell you.

So, how then can we give urban software a voice? How can we actually bring it to our attention in ways that it sorely deserves, but rarely gets? I want to argue, really, in four parts, which goes along with the paper.

- What I want to do, straight off, is to consider the preponderance of software in the city. How much of it is there, where is it, and what's it doing? A simple audit, nothing more than that.

- The second thing I want to do is to look at the way that software has been transmuted - the nature of software itself, of course, being changing, and as it does so, it produces new kinds of possibilities for action.

- Thirdly, we'll try and make some preliminary sense of what's going on with software, and that will mean in particular, visiting some fairly mundane programs like spreadsheets, to try to understand how, maybe, cities are being configured.

- Fourthly, we'll try to sum all of this up, although clearly, that's impossible to do in certain senses, at least we'll have a go. And in particular, in the knowledge that at this point in time, with the advent of various kinds of mobile technology, actually understanding what software is doing actually becomes even more important than it was.

Where Is Software?

Let's start with the first part of the paper. Let's try to understand just how much things might have changed, by going tack to the 1970's, the age of flared trousers and long hair. And what we will find there, is a whole series of ways at the time, that people were trying to

understand how machines themselves were actually agents with the world. And I want to harken back to a paper by Horvath. He tried to map out, quite literally, machine space, in cities. What he wanted to do was to show the way in which automobiles were almost taking over what was going on in cities. Increasingly, space in cities was actually given up to automobiles, to machines, and not to human beings. He clearly saw this in the most classical human terms.

Now, what I want to argue, is that actually, if you looked at Horvath's map, you could find, and actually do in certain circumstances, almost the same kind of map for software nowadays in the city. And yet, it brings no such question in its wake. And what I want to do is begin by simply auditing how software comes about.

Let's start with the humble automobile, which, after all, is one of those objects which is now becoming, more and more, loaded up precisely with software. And in the paper, we go through the whole series of ways in which that's happening, ending up with the strong possibility that before long, even in the next couple of years, what we will see, is that passenger cars will be connected with onboard computers, in various smart ways. And of course, these programs in automobiles are mirrored, are paralleled, by all sorts of other ways in which software makes its presence known within transportation systems. And in the paper, we simply go through all the intelligent initiatives of one form or another.

So that's the domain that Horvath was so keen on mapping out - the domain of the automobile. But we can move back from that to plenty of other places in the city, where software is alive and living. For instance, in the paper again, we look at elevators. An integral part of the infrastructure of cities, especially, of course, in New York, which have become increasingly software-rich. Held up often as an exemplar of a stupid machine, they're now becoming loaded with more computing power. Some people have argued that there's more computing in an elevator now than in the Apollo spacecraft, but that seems to be a favorite metaphor used in these kinds of things.

And then, we can go on to that, to all kinds of other, newer technologies at this time. In the paper again, we talk about the role of various kinds of smart, CCTV systems, which are moving their way around cities, as more and more cameras are put up, becoming better, or worse, depending on the way you look at what they do.

So these are just a few examples out of many, but is there a more general way of making an assessment of where software is in the city? The answer, of course - we wouldn't bother with the question otherwise - is 'yes.' And that's because of the millennium bug. And countries around the world have produced, in effect, audits of where software is in cities, as a result of the Y2K problem. Again, in the paper, we go through in some detail, the way in which all sorts of different audits were made as a result of the dangers, apparently, posed by the millennium bug.

What that did mean, of course, is that whatever else you can say, you now do have, for a snapshot of time, a whole series of audits for different kinds of cites, of the software that is in place within those cities, just at that particular point in time, though there are some caveats, that I'll make at the end.

Now, if up until now, this infusion of software into the city, has gone largely unnoticed, this isn't going to be true for much longer, and the reason for that, I'd argue, is that in the next ten years at least, what we're going to see is that mobile computing, and all kinds of technologies based on mobile telecommunication, are gradually going to make their way authoritatively, into the city. And they're going to do that, I think, in a general move that will become apparent everywhere to everyday people. It's hardly going to happen next week, but it is undoubtedly starting to happen. These technologies will permeate the physical world, invisibly interwoven with sensors, actuators, displays, and computational elements, embedded seamlessly into the everyday objects of our lives, and connected through a continuous network.

And this revolution is now starting to take place, with various kinds of third-generation technologies. One could talk for example about the way in which mobile phones are gradually becoming smarter and smarter, and more to the point, are becoming able to communicate with one another, and with all kinds of computing forms around and about. This will only be boosted by the advent of things like Bluetooth, which will mean, we can all be surrounded if we want to be with various kinds of mobile devices that can seamlessly communicate with each other.

Similar intelligent telecommunications systems are being developed in other ways. In the paper we into some detail, for example: wearable computers, and the way, in which, although they're crude at the moment, they're gradually going to become something that makes really quite a strong sense, and will in all sorts of ways, I think, become something that will be a part of our everyday lives. We talk about the way in which computation acts in textiles, which means that computing power gradually moves closer to the body and becomes part of everyday action.

I could go on and on about this, but let me go on about what this might mean in terms of other things which might be going on. Specifically, what this will mean is that cities will also become paces in which machines are communicating with each other without needing intermediaries. Communications technologies like Bluetooth will play a pivotal role in this, and one argument that can be made is that the number of phone calls between people will be overtaken by machines talking to machines on behalf of one another. These kinds of things are still in the future at the moment, but they're not nearly as far away as people have predicted.

Now, I wouldn't want to give the impression of technological finality, and in the paper, we go into some detail into the way that software is actually cut back by various kinds of different things about it. This means that it's not always a reliable technology; it's a patchy technology; it's a quirky technology, and so on. We talk in particular about programming languages, about distance, we talk about the way in which many programs work okay but do so against a background of what one might call 'ignorant expertise'. The argument has often been made that software is not designed particularly well. All I'm trying to say is that a lot of the time, programmers are producing programs that already, because they're modularized in one way or another, include elements that they don't understand themselves, let alone how the whole program is put together. This is, in a sense, a perennial thing that goes on, but whether it's a problem or not depends on the angle you take.

The next part of the paper really goes on from these particular points. And in conclusion to that section, all I really want to say is to remember that the Y2K orders show that we really don't know where a lot of software is being done. For example, the audit commission in the

U.K., in the end had to put in a disclaimer, because it really wasn't clear how many embedded systems there actually were in cites, where all of them were, and what some of them were actually doing. In a sense, they simply disappeared into the background.

The next section of the paper, which I will briefly go into, is essentially about the way in which computing programming itself has recently changed its character, with the advent of all kinds of new possibilities. What I do in that section - there really isn't time to talk about it - is I really go through the way in which new kinds of soft computing of various kinds are actually producing different kinds of styles by which software can be run, and different kinds of possibilities therefore, for software to be in the city, and do different kinds of things.

Software Acting Upon Cities

Let's talk about what it is that software might actually be doing in a city. And, to do this knowing that wherever we go in modern cities, we're being directed by software: driving in a car, stopping a red light, crossing the road, using an elevator, using the washing machine or the dishwasher, or the microwave, making a phone call, writing a letter, playing a computer game, and so one. What we hope we've established, at least, is the prevalence of software in cities, and how extraordinary it is that so much of this has been neglected, even though increasingly, these programs have come to run cities, in very strong ways, and to direct human bodies around them.

The way that we want to try and talk about this is to talk about a kind of governmentality, in the urban sense. One in which there is a new form of government coming about in cities, which works through four kinds of informationalization.

- First, in new forms of visibility,
- secondly in distinctive ways of thinking and perceiving,
- thirdly, specific ways of acting, intervening and directing, and
- fourthly, characteristic ways of making up to subjects.

We would argue that software is important on each of those dimensions. It contains characteristic forms of visibility, by informationalizing the city, and so producing new objects of analysis, it changes ways of thinking and questioning by producing new analytical procedures, it changes the nature of expertise, by producing new ways of making decisions, and it's changing the nature of human subjects, by producing enhanced capability. And by questioning not just what techniques the self consists of, but whether the self is actually a meaning happening at all. Software is therefore now becoming a key technology of government for modern states.

What I'm now going to do is, very briefly, is go through three particular ways that software intervenes in the government of cities, as it currently is, and as it undoubtedly will be. The first of these, is the most obvious, and I'm not going to talk about it today, because a lot of other people are going to talk about it. But it's the geography of software production - where, actually, software - code - is being written. There is work on this, though surprisingly little if you

look around, on the actual production of code itself, where all that writing is being written down. And in the paper, we summarize what literature there is and we also note the way in which there are two different kinds of geographies going on here.

One of them, if you like a conventional, industrial kind of geography, which involves hierarchies, which involves centers. The other involves things like open sources, which involves a much more general kind of geography - much less hierarchical than is often made out. That's the first way in which software intervenes in cites, and to me, an important one to look at in more detail. The point of production itself; where all this code is being produced.

Then secondly, there's the problem of what software consists of outside of all the little lines of code. What we want to argue is that it consists of rules of conduct able to be applied to a determined situation. These rules of conduct operate at a distance, so that too often, the code seems to have little to do with the situation to which it is applied. We would argue very strongly that there is a serious politics of software to be looked at much more strongly than it is at the moment. In a sense, software programmers, as they write the code, are often writing all sorts of rules of decision around the world, some of which are highly political, and often are passed over as being purely technical.

What ethnography of programmers there is tends to show the way that they will often see programming as a technical problem rather than a political problem. But what we argue in the paper is that code is certainly a way of making law, of a sort. It's a way of setting a whole series of stories framing a whole series of encounters. And these stories set out a lot of standards of conduct which can apply on all kinds of scales. We go into all these different ways in which that can actually happen, and in particular, the way that software works is to produce a whole series of classifications and standardizations to a situation, in ways which were formerly impulsive. Bowker and Star have a nice phrase for this, they call it the "categorical saturation" of the modern world.

It's very easy to talk about this in an abstract way, but it's actually relatively easy to see that software has actually had quite major effects of one form or another. And in the paper, looking at governmentality, the example we've chosen is the humble spreadsheet, something that people hardly look at all. It's there on the application, we do it, and that's really all there is to it. But it's worth remembering that low-cost spreadsheet programs only first appeared in 1979 with the introduction of VisiCalc. Within five years, one million spreadsheets were being sold annually worldwide, and they've then of course become more and more ubiquitous, leading to the success of programs like Lotus 1-2-3 and Microsoft Excel. Certainly, within ten years, it could already legitimately be claimed that businesses had become spreadsheet-conscious at various times. And subsequently, of course, spreadsheets have continued to evolve.

The spreadsheet has been able to be adapted so rapidly because its format mimicked the structure of the paper ledger. This is something that one often finds with software programs; they start looking like something already there, and then they transform into something that wasn't there before. What they do, of course, is they allow for all kinds of possibilities for calculating the world, which were not there before. They have become so common in many businesses, that they, in many ways, can be seen as a kind of language. Some people have argued that the true

language of some multi-national corporations is not some kind of German or Japanese, but actually that it's the spreadsheet.

You can see the way in which the spreadsheet has had an impact. To begin with, it provides a new opportunity for interaction - activity around the spreadsheet. Then it furnished management with all kinds of rhetorics they never had before. It asked you questions, and it furnished you with new kinds of information about companies which would not have been available before. In other words, spreadsheets create new stories of governance. New ways of running particular parts of businesses which were not available before and which have had impacts in all sorts of ways.

That's interesting in itself, but the other interesting thing is that, with this programming model, once it exists, can then migrate. One of the interesting things about spreadsheets is that they've moved out from simply being about financial calculation, and simply about business, into all other parts of the world as well. Just as an example, specialty spreadsheet programs are now available for applications as diverse as car purchasing, class notes and assignments, diving decompression, seed cultivation and trading, landfill gas production, archeological digs, laboratory management, chemical properties, music production, real estate, and so on and so forth. So they may have started in one domain, but soon the model moves elsewhere. A particular kind of software can migrate into all sorts of other kinds of domains, changing those as it goes.

In the paper, it may come across that spreadsheets are something negative. We're not particularly trying to suggest that; they can be, but it's also important to note the example that they have been used by artists, composers, and such in various ways, so it's by no means the case that that is so.

That brings us to the third and last of these points, and that is that the spread of software doesn't just, of course, have negative connotations. There are new forms of 'yes' as well as 'no' based in software. In the paper, we go into some of the ways that software is being used in the arts and humanities, and being used in all sorts of ways. For example, there are people here today from the MIT Media Lab. Certainly some of the things that the people from Media Lab have been trying to do is to find new expressive capacities for artists by actually producing new kinds of software packages. In paper we talk about some of those possibilities, for example, in the ideas of dance, and so on. So there are a whole series of ways that this can be a creative invention as well.

What I’m going to argue now is that you can see software in two different ways. One of them grand, and one of them not grand at all. In a sense, in an interesting way, it's probably a little of both. The problem, as I've tried to show, is that thinking about the place of software in cities, places made by, with, and from software, is actually a real problem. A problem because so little thinking is taking place around that kind of question. A part of, if you like, the taken-for-granted background. To some extent, it has been suggested that software will increase until it's a kind of virtual skin around human interaction. Now I'm sure it won't be that in quite the kind of science-fiction way that it's often argued. But it might be, in a much more mundane way. One way, then, of seeing software, is basically what it's doing - it's extending humanity in one form or another. In a sense, what we see, is an organism which is building more and more of itself

outside of itself. And software, if you like, is just the latest manifestation of that, but an important one of course, even so. The strings of words, if you like, or strings of codes, increasingly will be quite literally writing the world in various ways. And they will also be doing this even in terms of the way in which they get out into everyday environments like houses, and so forth.

What we will see, therefore, in your American society, is a kind of complex ecology of software, and increasingly what we'll see is a kind of ethology, which is rather like the Latour's work in Paris. Made up of a vast panorama, of signs, directives, and objects, which will be paralleled in the virtual realm, producing a very effective assembly of control.

That's at least one way of looking at it, and certainly one of the ways that people have interpreted software. Another way, which is rather more straightforward, is to take this as being rather too grand. After all, software is still written by human hands; it arises from striking a keyboard, the clicking of a mouse, the shifting of notes around on a desktop, of the consulting of manuals, to find out various standards and classifications, and so on. Software, still flows from the interface between body and object Though, what that interface is, precisely, is what many sociologists have been worrying about.

But maybe, instead of understanding software as maybe the next chapter in the evolution of humanity, and certainly, you can find books that will argue this, you can see it in a much more mundane way, as a practical extension of human interfaces, decrees and processes. The first is a simple extension of textuality. Modern western cities are, effectively, intertextual. From the myriad forms created by a bureaucracy, to the book, the newspaper, the web page, through the check-out till roll, the credit card slip, the letter, the email, the city, if you like, is one vast intertext, and the software is simply conjuring more and more of this intertext into existence. Cities are quite literally, being written.

Second, software is part of the paraphernalia of everyday urban life, revealed in one way or the other. It's one of those large technologies that are crucial to the bonding of urban time and space. The kind of technology that the same as the invention of the pencil, so mundane that you don't even think about it any more. Or the screw - a book's recently come out on that, that's why it's there - which literally holds cities together, in the strongest possible sense, and which, of course, have the result of going almost entirely unnoticed.

And then thirdly, people see software as a means of transport, as an intermediary passing information from one place to another. So, in those three cases, it may be possible to talk about most of what software is about, without having to go straight to hyperbole. Whatever the case, it's certainly the case that we can no longer go on thinking about it in the old, time-honored way. Five years, ago, I think that I would have muttered and moaned on about these kinds of things, and thought it highly unlikely that a lot of these things were going to take place. But I think now that the advent of software typifies the rise of new forms of technological politics, and new factors of political convention that were only just beginning to comprehend as political: politics of standards, classifications, and metrics. These orderings, written down as software, are, I think, becoming one of the chief ways of animating the city, and they should not be allowed to take us unaware, because there are serious politics to this, which we should be involved in.