Toolkits for the Mind

written by Jeremy Huggett | 17/04/2015

In 1975 the computer scientist Edsger Dijkstra wrote "The tools we use have a profound (and devious!) influence on our thinking habits, and, therefore, on our thinking abilities." Dijkstra was writing in relation to programming languages but the same might equally apply to the software products coded in those languages. In this, he recalls Marshall McLuhan's famous dictum: "We shape our tools, and thereafter our tools shape us" (although the pedant in me insists that this was actually by John Culkin).

The title for this post is shamelessly borrowed from an article by James Somers (2015) in which he seeks to argue that programming languages shape the way their users think:

> "Software developers as a species tend to be convinced that programming languages have a grip on the mind strong enough to change the way you approach problems—even to change which problems you think to solve."

Somers isn't alone in making this claim: for instance, David Berry argues for a need to understand program code and the systems so constructed since:

> "A close reading of code can … draw attention to the way in which code may encode particular values and norms … or drive particular political policies or practices." (2011, 9).

Similarly, Lev Manovich writes of how software appears to users:

> "… what *functions* it offers to create, share, reuse, mix, create, manage, share and communicate content … and *assumptions and models about a user, his/her needs, and society* encoded in these functions and their interface design" (2013, 29 – emphasis in original).

Writing about spreadsheets in 1984, Steven Levy argued:

> "The spreadsheet is a tool, and it is also a world view—reality by the numbers. If the perceptions of those who play a large part in shaping our world are shaped by spreadsheets, it is important that all of us understand what this tool can and cannot do."

Making things computable entails assumptions and beliefs at every level, from the deconstruction of a specific problem and its implementation in code by a programmer through to the end user's

understanding that the software performs the task they require. The programmer has different algorithms and options available to achieve a certain end, depending on their know-how and their prioritisation of the efficiency, reliability, elegance and aesthetics of the code. A user will generally be unaware of these alternatives or the implications of the chosen approach. Indeed, to some extent the relation of the user with the software has an element of circularity to it: unless we are able to access the code (assuming we were inclined or equipped to do so), we rely on the evidence of our experience – that what we put into the system comes back out again in a way in which we perceive as valuable or useful, hence we can believe that the software is performing in a more or less expected manner. Even if we are able to access the source code, the reality is that in all probability its scale and complexity makes it difficult even for experts to follow the ebb and flow of the code. Consequently, open source or not, we are frequently dealing with software black boxes where only the inputs and outputs are known to us.

So if there is agreement that the tools – whether the underlying code or the resulting software – are shaping us to some degree, does any of this really matter to archaeologists?

Well, if the tools we use affect our approaches past and present and potentially shape the future of the subject, and if we can't or won't get under the hood of these tools to understand how they work, then we cannot be sure about what impact they are (or are not) having. Software – and the underlying code – have all kinds of knowledge, theories, methodologies encoded within but rarely do these break through to the surface. Indeed, the software is often designed in such a way as to prevent this, and as Sherry Turkle pointed out some years ago (1997, 36-42) users are left in a position whereby we can see how something can be made to work rather than having knowledge of how something works. How often do we hack around with a piece of software until it (eventually) achieves more or less what we want, but we're still none the wiser about how? Introspection, in the sense of inspecting the running code, may be too ambitious, perhaps even not particularly useful, but introspection as a considered evaluation of the place, performance, and outcomes of a program is much more achievable.

For example, five years after the development of the first spreadsheet, VisiCalc, Steven Levy wrote of the way their introduction increased demand for quantitative decision making (not really a great surprise, perhaps), but also encouraged a 'what-if' approach and an emphasis on what could be represented numerically, leaving out the more intangible aspects of a problem (not unlike the spatial determinism associated with GIS). The formulae within spreadsheets are also hidden from sight – they can be seen (unless they are hidden and locked with a password) but aren't displayed by default, although it's not uncommon to receive complex spreadsheets stripped of the original formulae, obfuscating the origins of the calculated data. This makes understanding all the more problematic since we are unable to evaluate the underlying model and the assumptions embedded within it:

> "These formulas are based on assumptions made by the model maker. An assumption might be an educated guess about a complicated cause-and-effect relationship. It might also be a wild guess, or a dishonestly optimistic view." (Levy 1984).

For instance, I've commented elsewhere on the effect of the 'cut and paste' facility of word processors on writing. Similarly, what effect does the atomisation and structuring of data have on

the way we think about and subsequently interpret our data? What are the implications of the use of image-based modelling to record excavations? And what is the effect of using tools which have embedded within them modern spatial concepts and visual perspectives if the objective is to represent and understand past perceptions of environments and settings?

Amongst digital archaeologists, GIS has perhaps received the lion's share of this kind of introspection since the mid-1990s (for example, Hacıgüzeller 2012, Hu 2012) though there's much more to do. But few of the other software tools we use have received anything like the same attention. How have they shaped archaeological practice, and how are they continuing to reshape it? The few historiographies of archaeological computing that there are (most recently Djindjian 2015, Moscati 2015) tend to focus at a high level on people, organisations, and techniques. What we need are critical historiographies of the tools themselves.

References

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

F. Djindjian 2015 'Computers and Mathematics in Archaeology, Anatomy of an Ineluctable Success!', in F. Giligny, F. Djindjian, L. Costa, P. Moscati and S. Robert (eds.) *CAA2014: 21st Century Archaeology – Concepts, Methods and Tools* pp 1-6. (available at http://bit.ly/1FSCpmC).

P. Hacıgüzeller 2012 'GIS, critique, representation and beyond', *Journal of Social Archaeology* 12 (2), 245-263 http://jsa.sagepub.com/content/12/2/245

D. Hu 2012 'Advancing Theory? Landscape Archaeology and Geographical Information Systems', *Papers from the Institute of Archaeology* 21, 80-90
http://www.pia-journal.co.uk/article/view/pia.381/446

S. Levy 2014 (1984) 'A Spreadsheet Way of Knowledge'
https://medium.com/backchannel/a-spreadsheet-way-of-knowledge-8de60af7146e

L. Manovich 2013 *Software Takes Command*. Bloomsbury Academic.
http://issuu.com/bloomsburypublishing/docs/9781623566722_web

P. Moscati 2015 'Towards a History of Archaeological Computing: An Introduction', in F. Giligny, F. Djindjian, L. Costa, P. Moscati and S. Robert (eds.) *CAA2014: 21st Century Archaeology – Concepts, Methods and Tools* pp 9-15. (available at http://bit.ly/1FSCpmC).

J. Somers 2015 'Toolkits for the Mind', *MIT Technology Review*
http://www.technologyreview.com/review/536356/toolkits-for-the-mind/

S. Turkle 1997 *Life on the Screen: Identity in the Age of the Internet*, Simon & Schuster.