From Codework to Working Code: A Programmer's Approach to Digital Literacy

Bork, John
jrbork@wcnet.org
University of Central Florida

What does it mean to be digitally literate? Obviously it entails a basic familiarity with commonly used technologies, so that one may navigate the technological life world that has permeated nearly every aspect of the human one. One aspect of this knowledge is the recognition of computer languages, communications protocols, syntactic forms, passages of program code, and command line arguments, even when they have been taken out of their operational context for use as literary and rhetorical devices. In addition to the infiltration of the abbreviated language of email and text messaging into mainstream print media, it is now also commonplace to encounter programming keywords, symbols, operators, indentation, and pagination entwined with natural, non-technical, mother tongue expressions. *Codework* is the term associated with the literary and rhetorical practice of mixing human and computer languages (Hayles, 2004; Raley, 2002; Cramer, 2008). Types of codework span from intentionally arranged constructions intended for human consumption that do not execute on any real computer system, to valid expressions in bona fide programming languages that are meaningful to both human and machine readers. Examples of the former include the work of Mez (Mary-Anne Breeze) and Talon Memmott, of the latter, the work of John Cayley and Grahan Harwood (Raley, 2002; Fuller, 2008). Rita Raley notes, however, that of the popular electronic literature of the early twenty first century, there is "less code per se than the language of code." In addition to its infusion for literary effect, program source code may be cited in scholarly texts like conventional citations to explain a point in an argument. Although it is more common to encounter screen shots of user interfaces, examples of working source code appear on occasion in humanities scholarship. This study will briefly consider why working code has been largely shunned in most academic discourse, and then identify the types and uses of bone fide code that do appear, or are beginning to appear, in humanities scholarship. Its goal is to suggest ways in which working code – understood both as code that *works*, and as the practice of *working* code – plays a crucial role in facilitating digital literacy among social critics and humanities scholars, and demonstrate through a number of examples how this effect may be achieved.

The first argument in favor of studying computer code in the context of humanities scholarship can be drawn from N. Katherine Hayles' methodological tool of Media-Specific Analysis (MSA). Probing the differences between electronic and print media when considering the same term, such as hypertext, requires comprehension of the precise vocabulary of the electronic technologies involved. A second, more obvious argument comes from the growing disciplines of Software Studies and Critical Code Studies. If critical analysis of software systems is to reveal implicit social and cultural features, reading and writing program code must be a basic requirement of the discipline (Fuller, 2008; Mateas, 2005; Wardrip-Fruin, 2009). As the media theorist Friedrich Kittler points out, the very concept of what code is has undergone radical transformations from its early use by Roman emperors as cipher to a generic tag for the languages of machines and technological systems in general; "technology puts code into the practice of realities, that is to say: it encodes the world" (Kittler, 2008). Or, following the title of Lev Manovich's new, downloadable book, software takes command. Yet both Kittler and Manovich express ambivalence towards actually examining program code in scholarly work. A third argument, which will form the focus of this study, is reached by considering the phenomenon of *technological* concretization within computer systems and individual software applications. According to Andrew Feenberg, this term, articulated by Gilbert Simondon, describes the way "technology evolves through such elegant condensations aimed at achieving functional compatibilities" by designing products so that each part serves multiple purposes simultaneously (Feenberg, 1999). The problem is that, from the perspective of a mature technology, every design decision appears to have been made from neutral principles of efficiency and optimization, whereas historical studies reveal the interests and aspirations of multiple groups of actors intersecting in design decisions, so that the evolution of a product appears much more contingent and influenced by vested interests. The long history of such concretizations can be viewed like the variegated sedimentation in geological formations, so that, with careful study, the outline of a technological unconscious can be recovered. The hope is that, through discovering these concealed features of technological design, the the unequal distribution of power among social groups can be remedied. Feenberg's project of democratic rationalization responds to the implicit oppression of excluded groups and values in technological systems by mobilizing workers, consumers, and volunteers to make small inroads into the bureaucratic, industrial, corporate decision making.

For computer technology in particular, digital literacy is the critical skill for connecting humanities studies as an input to democratic rationalizations as an output. Working code replaces the psychoanalytic session for probing the technological unconscious to offer tactics for freeing the convention-bound knowledge worker and high tech consumer alike. Many theorists have already identified the free, open source software (FOSS) community as an active site for both in depth software studies and for rich examples of democratic rationalizations (Fuller, 2008; Yuill, 2008; Jesiek, 2003). Simon Yuill in particular elaborates the importance of revision control software for capturing and cataloging the history of changes in software projects. As as corollary to this point, it can be argued that concealed within these iterations of source code are the concretizations that make up the current, polished version of the program that is distributed for consumption by the end users, and from which the technological unconscious may be interpreted. However, even when they are freely available to peruse in public, web-accessible repositories, these histories are only visible to those who can understand the programming languages in which they are written. Therefore, it is imperative that humanities scholars who wish to critically examine computer technology for its social and cultural underpinnings include working code - as practicing programming - in their digital literacy curricula.

References

Cramer, Florian (2008). 'Language'. *Software Studies: A Lexicon.* Fuller, Mattew (ed.). Cambridge, Mass: The MIT Press, pp. 168-74.

Feenberg, Andrew (1999). *Questioning Technology.* New York: Routledge.

Fuller, Matthew (2008). 'Introduction'. *Software Studies: A Lexicon.* Fuller, Mattew (ed.). Cambridge, Mass: The MIT Press, pp. 1-13.

Hayles, N. Katherine (2004). 'Print is flat, code is deep: the importance of media-specific analysis'. *Poetics Today.* **25(1)**: 67-90.

Jesiek, Brent K. (2003). 'Democratizing Software: Open Source, the Hacker Ethic, and Beyond'. *First Monday.* **8(10)** (accessed 5 October 2008).

Kittler, Friedrich (2008). 'Code'. *Software Studies: A Lexicon.* Fuller, Mattew (ed.). Cambridge, Mass: The MIT Press, pp. 40-7.

Mateas, Michael (2005). 'Procedural literacy: educating the new media practitioner'. *On The Horizon. Special Issue. Future of Games, Simulations and Interactive Media in Learning Contexts.* **13(1)** (accessed 21 October 2009).

Raley, Rita (2002) (8 September 2002). 'Interferences: [Net.Writing] and the practice of codework'. *Electronic Book Review.* (accessed 7 October 2009).

Wardrip-Fruin, Noah (2009). *Expressive Processing: Digital Fictions, Computer Games, and Software Studies.* Cambridge, MA: The MIT Press.

Yuill, Simon (2008). 'Concurrent version system'. *Software Studies: A Lexicon.* Fuller, Mattew (ed.). Cambridge, Mass: The MIT Press, pp. 64-9.