SOFTWARE STUDIES

By Matthew Fuller, ed. (forthcoming)

Reviewed by Anne Helmond
June 2007, University of Amsterdam

Software Studies, a forthcoming lexicon edited by Matthew Fuller, consists of thirty-nine entries from mostly different authors. The title refers both to the object of study and the form of the project consisting of numerous short studies. Each of the "software studies" in the book stands on its own and Fuller celebrates the multi-disciplinary diversity of the authors. They come from different fields of study including art and design, literary theory, computation and free and open source software. Fuller has not gone as far as to attempt to start a new field of study but instead *Software Studies* calls for new theorizations of software from areas that "have not historically 'owned' software" such as media studies. The fields that are currently concerned with culture and media could contribute to a new approach to software with their critical perspectives on politics, society and matter.

Fuller states that he has chosen the form of a lexicon because it is provisional, scalable and contains pathways. It is provisional because it serves for the time being because software is not a static object and is therefore hard to pin down. Relations in and around software are constantly changing and a lexicon can serve as a temporary overview. Unlike a dictionary a lexicon is scalable and it does not strive to be complete and this incompleteness is "a virtue" according to Fuller. The entries can be seen as different pathways into software that do not strive to depict a whole. Connections between these pathways are made by the various authors but can also be constituted by the reader itself. In short, both software studies and the *Software Studies* lexicon can be seen as a specific approach whereby each entry is a pathway into thinking about software.

This approach does not seem to have a shared methodology except for creating new pathways into software. This could be due to the different backgrounds of the authors who all contribute to the discourse of software studies from their own

perspective and paradigm. This is both the strength and weakness of *Software Studies*: at times the lexicon seems uneven with an overemphasis on computing which is a neglected aspect of software according to Fuller. However, in order to bring back this neglected aspect too many entries fall in the lexicon back on Turing et. al.

Software Studies builds on Fuller's previous works *Behind the Blip* and, to a lesser extent, *Media Ecologies*. *Behind the Blip* consists of several essays on the topic of the culture of software. In the opening essay of *Behind the Blip* Fuller argues for a "software criticism" that moves authors writing about computers away from the performance of software towards a more critical approach. This new critical approach is not concerned with detailing the functionality of a particular piece of software but is rather concerned with the question how software consists of different elements that are embedded in a dynamic web of relations.

An excellent example of this critical approach is the essay 'It Looks Like You Are Writing A Letter' that critiques the popular word-processing software Microsoft Word revealing that "software constructs sensoriums, that each piece of software constructs ways of seeing, knowing and doing in the world at once contain a model of that part of the world it ostensibly pertains to and that also shape it every time it is used." Fuller argues that software can be seen as a synthesis, a form of amalgamation or assemblage, of different layers that do not imply a static whole. This dynamic synthesis is also the subject of Fuller's book *Media Ecologies* that uses a materialistic approach to identify three forces of objects in media ecologies: affordances, material substrates and memes. In a sense, software is described as having a vitality; it derives its energy from these forces that cause collision, (dis)connection and interaction underwriting their unstable and dynamic nature.

Lev Manovich previously addressed the importance of studying software in *The Language of New Media* in 2001. He states that media have become programmable and that we need a new field of study to address the issues that arise from this turn in our culture. Not only has software quietly penetrated our daily life but it has also become invisible. The ubiquity and so-called transparency of software renders it invisible but at the same time it points out the importance of studying it. Manovich has studied software from a formalist approach by taking terms and categories from computer science and applying them to new media that have become programmable. According to Manovich five principles distinguish new media from the older media namely, numerical representation, modularity, automation, variability and transcoding (27-48). These principles point to the relation that new media and

software have with code.

Adrian Mackenzie takes issue with Manovich with an interesting take on code and software in *Cutting Code: Software and Sociality* (2006) and notes that software is a very mutable object that is entangled in a web of relations. Mackenzie sees software as a social object and process that is intrinsically linked to code as a material and practice. He points to the problems of Manovich's formal analysis because it abstracts software from practices and contexts surrounding coding and reduces it to "relations and operations (such as sorting, comparing, copying, removing) on items of data." (Mackenzie 2006) These relations and operations are seen as quite stable forms and are often directly transferred from the field of computer science. Instead of abstracting and formalizing software Mackenzie argues for an ontology of software that deals with its mutability. This mutability arises from the agential relations indexed by code of the social web that software weaves. Mackenzie, as one of the authors of the *Software Studies* lexicon, contributes to software studies by arguing that we should render software visible and notice the agency it provides, generates and distributes:

> At stake here is an account of software as a highly involuted, historically media-specific distribution of agency. This account diverges from a general sociology of technology in highlighting the historical, material specificity of code as a labile, shifting nexus of relations, forms and practices. It regards software formally as a set of permutable distributions of agency between people, machines and contemporary symbolic environments carried as code. Code itself is structured as a distribution of agency. (Mackenzie, 19)

So what is next for the field of software studies? After having finished reading *Software Studies* it has not become quite clear what is to be done since it does not provide a unified approach or methodology. However, the lexicon is an excellent starting point for those who wish to be introduced into software studies. But what about those who wish to contribute? Even though a lexicon is provisional, scalable and offers pathways, in a printed form it still implies some kind of a finished whole. Software as a shifting nexus of relations is in a constant flux and pathways into software may disappear, change or be added. The print form is not fit to adjust to such changes because once the text has been printed there is no way to adjust it. Revising is a possible solution for this problem. Recently, the famous lexicon *Keywords* by Raymond Williams has been revised by several editors resulting in *New Keywords: a revised vocabulary of culture and society*. Original entries of the lexicon were updated and new entries were added twenty five years after the original publication.

A more fitting solution for *Software Studies* would be supplying a digital

environment in which changes in entries can be made without losing the original entry and can be tracked. Such environments are currently known as wikis which might be an ideal work form for software studies. Like software, wikis are often seen as a shifting nexus of relations that contain provisional, scalable pathways into other topics. A wiki, or another hypertextual environment might render the different pathways into software and their connections more visible thus expanding the knowledge about software and its relations. To perceive a better understanding of software we need to create more pathways. Would software studies benefit from using software to create these pathways and write about its studies?

References

Fuller, Matthew. *Behind the Blip: Essays on the Culture of Software*. Brooklyn, USA: Autonomedia, 2003.

Fuller, Matthew, ed. *Software Studies*. Cambridge, USA: The MIT Press (forthcoming).

Mackenzie, Adrian. *Cutting Code: Software And Sociality*. New York, USA: Peter Lang Pub Inc., 2006.

Manovich, Lev. *The Language of New Media*. Cambridge, USA: The MIT Press, 2002.