Understanding software development in practice

Suenson, Espen 2015. How Computer Programmers Work – Understanding Software Development in Practice. Turku: Department of Information Technologies, Åbo Akademi. 287 pp. Diss. III. ISBN 978-952-12-3225-1. ISSN 1239-1883.

During the last few decades, computer programming and software development have become important practices in societies more or less all over the world. New stakeholders and organisations have risen and software now underpins infrastructures, services and products from such diverse areas as avionics and farming to entertainment and gaming. How to understand the development of products within these areas and the practices that are taking place has been the aim of Espen Suenson's doctoral thesis, *How Computer Programmers Work – Understanding Software Development in Practice*. The thesis is framed as a scholarly combination of software engineering and European ethnology, making it a hybrid thesis, which has been rather uncommon in the academic world. Usually academic theses and dissertations are evaluated and assessed within the framework of separate disciplines. In this case, the work has a cross-disciplinary scope, and the form of the thesis has given rise to new opportunities but also a few challenges.

Software as a concept and as a phenomenon has been receiving attention from scholars within the social and cultural sciences for some time now. Often the approaches have been cross-disciplinary in nature, drawing on theoretical and methodological traditions from various areas. Early forerunners like Marshall McLuhan and Friedrich Kittler have inspired bold and somewhat speculative scholarly endeavours. Several studies have been related to upcoming fields like science and technology studies, media studies or software studies, with the last of these fields being epitomised by the work of such scholars as Matthew Fuller and Lev Manovich. Suenson mentions some of these scholarly works in his book, but then decides to take another research route.

Suenson's ambition has been to broaden the scopes of the disciplines of ethnology and software engineering by embracing a quite narrow theoretical aspect of ethnology, combining it with traditional hermeneutical theory and applying it to the practices of software development. This ambition to broaden the horizons and scopes of several disciplines, as well as the very form of the thesis, does produce some challenges. It requires an open-minded reading and approach, one which overlooks the fact that Suenson does not analyse or problematise how several social dimensions play out in his research material. This makes the work quite unusual among other recent ethnological studies on the workplaces, technology and professional practices. But Suenson's book also offers rewards, through which hybrid knowledge is produced and exchanged in unexpected ways.

Methodologically, Suenson has chosen not to use theory in order to form new concepts and develop novel theoretical approaches. Instead, he applies cultural theory and hermeneutical theory to his empirical material in a way that to some extent is similar to compatibility tests of the world of programming. This is a logic through which some selected theories are tested to see if they work together with a specific empirical material.

With the help of Hans-Georg Gadamer's theories and philosophical works on hermeneutics and rhetorics, as well as cultural form theory inspired mainly by the Danish scholar Thomas Højrup, he deals with several specific dimensions of computer programming practices. Throughout the thesis, Suenson tackles the knowledge gap between the ways that the production of software has been described and analysed and what is really going on in the work practices involving computer programming and software development. The aim of the thesis is not to directly change or improve the studied programming practices, but instead to facilitate a better understanding of and better ways to describe 'how computer programmers work'.

The book is based on a comparative analysis of two very different forms of programming practices, namely game programming and safety critical programming. A smaller game development company called Tribeflame is analysed as well as larger enterprises working with safety critical programming. The latter kind of programming is used when malfunctions and failures pertaining to the produced software could have very serious consequences, which requires certain procedures of assessment and control during the development process.

One strength of the book is the way that programming practice is discussed in the case studies, especially in the parts about Tribeflame. Chapter five on game programming and Tribeflame is an ethnographically interesting part of the work. Here, Suenson's ethnographic method becomes concrete, and the physical and material context of programming practices is presented. Aspects of programming practices within this empirical field are highlighted and scrutinised. From an ethnological and ethnographic point of view, the contexts, situations and sites of this specific industry could, however, have been even more thoroughly examined. It would also have been interesting to include more ethnographic material from the world of safety critical programming.

Suenson make comparisons between the companies and industries, but especially the more detailed ethnographic material from Tribeflame tends to disappear among general assumptions about game programming. Some more thorough discussions and comparisons of, including more ethnographic details regarding, the very different fields of software production examined in this book would have made the work stronger from an ethnological standpoint. Suenson mentions that game development is connected to other creative practices. But he provides no further comparisons or analyses of this industrial context. Instead, game programming is related to safety critical programming in totally different industrial contexts. With more discussion based on ethnographic details and comparisons, he would have provided readers with a better understanding of the specific programming practices of companies like Tribeflame. For example, he stops the discussion after stating that it is important for game developers to make products that are 'fun' and that companies working with safety critical programming have to develop 'safe' products. Our ethnological understanding of software development would have benefitted if he had scrutinised in more detail what is really meant by 'fun' and 'safe' and how these concepts are manifested in everyday programming practices in different industries. In the present analyses, notions of 'fun' and 'safe' become secondary to the companies' primary aim, which according to Suenson is to make 'their products useful' (p. 203).

Since digital culture and practices, such as computer programming, are spreading on a huge scale more or less globally today, it is important for studies to test new combinations and take new roads to knowledge. Suenson has started to tread one of these roads. The end of the road is still somewhere beyond the horizon. However, at the beginning of the book Suenson clearly states his specific goal for the thesis, and he reaches it. This book contributes to our knowledge about software programming in an interesting way, but there is still much work to be done and many roads for researchers to tread in order to better understand the implications of software and how it is designed, developed, utilised, imagined and transformed.

Robert Willim