Chapter 2.14
Deconstructive Design as an Approach for Opening Trading Zones

Doris Allhutter
Austrian Academy of Sciences, Austria

Roswitha Hofmann
WU Vienna, Austria

ABSTRACT

This chapter presents a critical approach to software development that implements reflective competences in software engineering teams. It is grounded within qualitative research on software engineering and critical design practice and presents conceptual work in progress. Software development is a socio-technological process of negotiation that requires mediation of different approaches. Research on the co-construction of society and technology and on the social shaping of technological artefacts and processes has highlighted social dimensions such as gender and diversity discourses that implicitly inform development practices. To help design teams implement reflective competences in this area, the authors introduce 'deconstructive design'—a critical-design approach that uses deconstruction as a tool to disclose collective processes of meaning construction. For this purpose, the idea of value-sensitive design is linked to approaches of practice-based, situated and context-sensitive learning and to the concepts of 'trading zones' and 'boundary objects'.

INTRODUCTION

The rich research tradition on the social shaping of technology (e.g. Bijker & Pinch, 1984; McKenzie & Wajcman, 1999; Nørbjerg & Kraft, 2002) and on the co-construction of society and technology (e.g. Bowker & Star, 1999; Rip, Misa, & Schot, 1995; Suchman, 2007) illustrates software development as a socio-technological process in various ways. Firstly, it takes place within organisations and therefore system specifica-

DOI: 10.4018/978-1-61350-456-7.ch2.14

tions and their implementation are co-determined by the organisational setting in which they are developed (such as organisational structures, engineering cultures of the respective sector, or work practices) (see Dittrich, Rönkkö, Eriksson, Hansson, & Lindeberg, 2008). Secondly, design decisions—although mediated by methods and tools of software engineering—represent the outcome of processes of negotiation and meaning construction (e.g. Akrich, 1995; Floyd, Züllighoven, Budde, & Keil-Slawik, 1992); in this sense everyday knowledge and social discourses become operative in the development process as hidden assumptions and belief-systems. The described 'situatedness' of software engineering in organisational and social contexts raises the question of how structures such as in/formal hierarchies and discursive hegemonies reproduced in everyday practices affect development processes and design decisions (see also Suchman, Blomberg, Orr, & Trigg, 1999).

Grounded in qualitative software engineering research and critical design practice this paper presents conceptual work in progress. It suggests a critical approach to software design that sustainably implements reflective competences in software development teams. We build on previous research that resulted in specifying a collective discourse-analytical method for this purpose (Allhutter, Hanappi-Egger, & John, 2007). This method is called mind scripting and allows one to make visible societal discourses and hidden sense-making that unconsciously shapes system design. On the basis of the above mentioned research tradition, we consider software development as a situated, social process of negotiation requiring the mediation of different viewpoints and approaches (such as views of managers, system designers, graphical and sound designers, programmers). Moreover, design decisions are always based on commonly held beliefs on social contexts. Gender discourse serves as a useful example to illustrate how socio-technological practices emerge from cultural processes of negotiation and meaning construction. The preceding research has shown how implicit discourses on gender and other diversity factors such as age, ethnicity or sexuality provide social meaning to seemingly technology-centred design decisions. In this contribution, we elaborate a 'deconstructive design' approach and highlight how mind scripting can gain from approaches to practice-based, situated and context-sensitive learning (Lave & Wenger, 1991) and from the concepts of 'trading zones' (Kellogg, Orlikowski, & Yates, 2006) and 'boundary objects' (Star & Griesemer, 1989). 'Deconstructive design' aims at reflecting collective work practices which unconsciously reproduce hegemonic discourses and by doing this narrow the spaces of innovation. The approach encourages the sustainable implementation of reflective practices. Essentially, 'deconstructive design' not only aims at developing value-sensitive software or artefacts that embody cultural critique (Dunne & Raby, 2001) but it goes beyond this objective. It is meant to enable sustainable practice-based learning in two respects: Firstly, by building competences in the reflective process, it paves a way for implementing constant process improvement. Secondly, by building reflective gender and diversity competences, it inspires value-sensitive innovation.

The guiding questions for our theoretical endeavour are: How can qualitative software-engineering research benefit from approaches to situated and context-sensitive learning? How can software design teams and their organizations broaden their scope of professional action by identifying 'boundary objects' and 'trading zones' in their everyday work routines? How can 'deconstructive design' sustainably foster reflective competences of design teams by making negotiable value-related decisions and development practices? To deal with these issues the paper is organised as follows: The following section locates our research within qualitative software-engineering research and briefly outlines the theoretical underpinning of our deconstructivist