A DECONSTRUCTIVIST METHODOLOGY FOR SOFTWARE ENGINEERING

Doris Allhutter
Institute of Technology Assessment, Austrian Academy of Sciences, Strohgasse 45/5, Vienna, Austria

Keywords: Co-construction, Collective work practices, Theories-in-use, Deconstruction, Situated learning, Trading zones.

Abstract: Grounded within qualitative research on software engineering and science and technology studies, the paper introduces a deconstructivist methodology for software engineering. Software engineering is a socio-technological process of negotiation embedded in organizational and societal contexts. Thus, social dimensions such as hidden assumptions of use contexts (e.g. based on diversity aspects such as age, gender, class or cultural diversity) implicitly inform development practices. To foster reflective competences in this area, the paper suggests using deconstruction as a tool to disclose collective processes of meaning construction. For this purpose, the idea of introducing a deconstructive process to software engineering is linked to approaches of practice-based, situated and context-sensitive learning.

1 INTRODUCTION

This paper is conceptual work in progress and deals with the question of how in/formal hierarchies and discursive hegemonies reproduced in everyday work practices affect software design processes. It presents a discourse-theoretical, deconstructivist approach to software engineering that sustainably implements practice-based learning processes in development teams. Theoretically based in approaches to the co-construction of society and technology, it connects qualitative software engineering research and process improvement with science and technology studies (STS), critical technical practice and organisational learning.

Research on the co-construction of society and technology describes software engineering as a socio-technological process: system specifications and their implementation are co-determined by the organisational setting in which they are developed (such as organisational structures, engineering cultures, or work practices) (e.g. Rip, et al., 1995); design decisions—although mediated by methods and tools of software engineering—represent the outcome of processes of negotiation and meaning construction; in this sense everyday knowledge and social discourses become operative in the development process as hidden assumptions and belief systems. For example, Akrich (1995) suggests that engineers anticipate the interests, skills, and behaviour of future users. They objectify these user representations in the developed artefacts and thereby attribute specific competencies, actions and responsibilities to users and artefacts.

Scholars in STS have highly contributed to strengthen the understanding of how societal discourses and everyday work practices guide design projects and pre-structure use contexts. They have argued with the inscription of societal discourses to software artefacts, to key concepts of computer science as well as to processes and methods of software engineering (e.g. McKenzie & Wajcman, 1999; Oudshoorn & Pinch, 2003; Suchman, 2007). However, the socio-political orientation of STS has only recently raised a growing demand for actual intervention into technological development processes.

Vice versa—in order to account for a conception of software engineering as socio-technological activity—scholars of applied informatics advocate the need for analytical approaches integrating socio-scientific, qualitative methods. Whereas empirical research in computer supported cooperative work (CSCW) has a tradition of using qualitative approaches, also scholars in software engineering (SE) and information systems (IS) have begun to use qualitative methods for analysing and reflecting development practices (see Dittrich, et al., 2009; Trauth, 2001). As Dittrich, et al. (2007, p.355) have

Allhutter D. (2010).
A DECONSTRUCTIVIST METHODOLOGY FOR SOFTWARE ENGINEERING.
In *Proceedings of the Fifth International Conference on Evaluation of Novel Approaches to Software Engineering*, pages 207-213
DOI: 10.5220/0003001502070213

summarized, SE researchers investigating the influence of deploying specific methods on the outcome of an engineering process use qualitative methods merely for generating hypotheses, which then serve to identify quantifiable relationships between methods and outcome. As an exception, the sub-field of requirements engineering is more attentive to qualitative methods. The intervention-oriented IS discourse seeks to initiate process improvement and quantitatively and qualitatively to evaluate the implemented measures. Only a few approaches combine qualitative research with the improvement of development methods and processes (e.g. Dittrich, et al., 2008).

Critical technical practice suggests methodological approaches developed to integrate social and critical theory into technological design. Design approaches, such as participatory design, value-sensitive and reflective approaches combine analytical and interventionist objectives (e.g. Mathiassen, 1998; Sengers, et al., 2005). Especially the latter approaches integrate a reflection of work practices as an essential part of systems development. As regards the reflection of societal questions, 'value-sensitive design', introduced by Friedman, et al. (2006), suggests putting values such as human justice and welfare in the centre.

Whereas the mentioned value-related approaches aim at modifying the vision of technical actors to consider diverging interest and predefined societal values, I argue that it is crucial to focus on reflecting cooperative work practices that (re-)constitute social meaning and thus hegemonic values and societal structures. This means, that engineering teams need to consider the hidden normativity of their development practices and ad-hoc decisions and reflect on how these co-shape engineering processes. Thereby, the focus shifts from balancing diverging interests to negotiating who (in/formal hierarchies) and what (discursive hegemonies) is given normative power on the basis of which values. In pursuing this objective, I focus on diversity-related questions with regard to user-centred design. Whereas these have been an important issue in STS, applied research has not yet to a large extend taken into account the ongoing discursive construction of social categories of diversity (such as class, age, cultural diversity, gender). A discourse-theoretical approach considers software engineering as a situated, social process of negotiation requiring the mediation of different viewpoints and approaches. In this spirit, design decisions are always based on commonly held beliefs and assumptions on societal contexts; these become manifest in theories-in-use, i.e. discourses implicitly guiding work practices.

The presented methodology integrates approaches to practice-based, situated and context-sensitive learning. Contrary to well-known knowledge management concepts which aim at the provision and use of explicit and implicit expert knowledge (e.g. Nonaka & Takeuchi, 1995) these approaches focus on theories-in-use and their underlying belief systems. The paper is organized as follows: Section 2 provides the theoretical and methodological basis; section 3 introduces the key concepts of the deconstructivist approach, which is outlined in section 4; the paper closes with an outlook to future work.

2 CO-CONSTRUCTION AND DECONSTRUCTIVISM IN SE

The notion of software engineering as a social, multidimensional process of negotiation emphasises the role of implicit everyday theories and societal discourses in engineering processes. Although the mentioned research fields, in one way or another frequently refer to the notion of co-construction or 'social dimensions' of software engineering, it still remains unclear how to grasp societal aspects in engineering practice and which to consider. Software development requires communication across professional boundaries and often also cooperation across decentred working environments. This indicates that diversity relations—within the engineering team and in its discourses—are crucial success factors. Whereas, as a consequence, diversity management has gained in importance in multinational enterprises, the label 'diversity' often remains obscured with regard to users or activates ideas about, for example, culture specific or gender specific requirements, attributing special (and often stereotypical) needs and preferences to particular user groups. Diversity-related discourses are a central issue in researching how socio-technological practices emerge from cultural processes of negotiation and meaning construction. However, design practice, and interestingly even user-centred design, does hardly make use of diversity approaches. I suggest that building reflective competences and organising learning within development projects in this respect has a twofold impact: Firstly, taking account of the diversity of practitioners in terms of their different educational backgrounds and professional self-conceptions enables them to tap the full potential available within the team, whereas

commonly hegemonic perspectives are privileged over alternative ways of knowing. Secondly, learning how user representations and linked societal discourses silently guide specification and implementation practices allows to overcome the pitfalls of stereotyping and thereby to gain capacities for action that are commonly narrowed down by hegemonic views.

The research objective of investigating how hidden theories-in-use and organisational discourses translate into collective meaning constructions within design processes clearly poses a methodological problem. Qualitative SE researchers have suggested a variety of approaches such as ethnomethodology (Rönkkö, 2007), ethnographically informed case studies (Robinson, et al., 2007), and grounded theory (Coleman & O'Connor, 2007). They predominantly resort to methods like interviews, focus group discussions, participatory observation, and analysis of communication and design documents. In their study on the lack of learning in software engineering teams McAvoy and Butler (2007) discuss the problem of how to research hidden phenomena that even those involved are reluctant to approve of or even unaware of due to cognitive discrepancies. The authors suggest that participatory observation is an appropriate method for disclosing hidden or ignored differences between 'espoused theories' and theories-in-use. Whereas participatory observation is adequate for researching established 'ways of doing', it does not reveal the underlying processes of attributing meaning to them—and thus does not enable a team to reflect on their adequateness. Therefore, I suggest that for researching the social constructedness of, for example, design decisions and their underlying belief systems and sense-making, a deconstructivist approach is most abundant. Based in discourse theory, deconstructivist methodologies focus on the reproduction of power by tracing the performativity of discourses. As Foucault (1971) and Butler (1990) have illustrated, powerful societal discourses, non-discursive practices and the objectification of these discourses and practices knit together and thereby (re-)produce societal hegemonies and power relations. Deconstruction questions the normativity of discourses and practices by revealing the constructedness of seemingly 'natural' sense-making; it aims at denaturalizing self-evident causalities which implicitly inform meaning constructions.

As Dittrich, et al. (2008, p.236) explain, established 'ways of doing' are produced and re-produced through collective and mutually intelligible practices. They are understandable as meaningful behaviour with respect to a common frame of reference and provide a base for ad hoc reactions to situational contingencies. Reflecting work practices and their base of implicit knowledge implies disclosing the pre-structuring mechanism of in/formal organisational hierarchies and hegemonic discourses (see Allhutter & Hofmann 2010). Such kind of investigative reflexivity demands maintaining a critical distance to one's own practices of constructing meaning. For this purpose, Allhutter and Hanappi-Egger (2005) suggest the method of 'mind scripting': As elaborated in Allhutter (in review), this analytical method is applied to trace the interconnectedness of the engineers' professional know-how with practical ways of problem-solving that are co-shaped by societal implications and discourses. It enquires how developers appropriate social structures, everyday experiences and educational and professional backgrounds and how these collective subjectivation processes translate into inherent professional self-conceptions and work practices that eventually materialise in software artefacts. At the same time, 'mind scripting' is a practicable tool for a team to deconstruct and reflect on its established practices. Thereby, hidden belief systems that reproduce dominant viewpoints are disclosed and underlying values are challenged.

3 SITUATED (UN) LEARNING

Argyris (2002) has described different levels of learning: Single-loop learning asks whether we are doing things right; double-loop learning includes questioning the underlying assumed causality and addresses the question of whether we are doing the right things. Triple-loop learning, a concept introduced by Flood and Romm (1996) adds a third loop that asks 'Is rightness buttressed by mightiness and vice versa?' and thus questions underlying value systems. The repeated questioning of learning routines is an important prerequisite for sustainable learning. Integrating all three levels of learning should initiate power-critical and change-oriented reflection processes in organisations. The notion of 'mightiness' may refer to formal and informal organisational hierarchies and, as Allhutter and Hofmann (2010) add, to hegemonic discourses or practices. In systems development providing functionality and the adequateness of the specification (double-loop learning) are well-established activities; the third loop, i.e. questioning implicit assumptions and value systems, is not yet included but is

important to prevent the implementation of very specific perspectives (Hanappi-Egger, 2006).

Making use of the concept of triple-loop learning in the context of a deconstructivist approach to software engineering also puts forward the need to reflect learning processes in terms of structures and discursive patterns of engineering teams. What we are learning from experiences and latently guiding discourses is deeply inscribed in embodied everyday practices and our cultural beliefs and value systems. At the same time, this means that learning as a social practice strongly relies on processes of unlearning of implicit sense-making and of consciously re-negotiating meaning (see Hedberg, 1981). As mentioned, theories-in-use may unconsciously rely on societal hegemonies and power relations to exclude alternative ways of knowing or to incite stereotypical assumptions. As regards diversity, the triple-loop learning perspective leads to the following questions: Which social assumptions from specific societal discourses do developers access in the design process? Are these assumptions perceived as right because they are legitimated by in/formal hierarchies and by hegemonic value-systems?

An extremely valuable approach in this respect has been provided by Lave and Wenger (1991) whose concept of 'situated learning' uses triple-loop learning to try and guide cooperation across professional boundaries. The authors conceptualise learning as a social process of participation in communities of practice. Communities of practice are groups of people who share a domain of interest. The members of such communities create relationships in order to share information, resources and experiences. Communities of practice are learning networks or thematic groups that are not limited by formal structures or organizational boundaries. They are important for knowledge creation in and between organizations and for the emergence of learning opportunities that are linked to performance. By focussing on such communities, the authors conceptualise learning as a process informed by societal structures and identity constructions that help to identify power structures as an essential factor for learning. Also from this perspective, diversity relations—within the organization itself and in its discourses—can be highlighted as crucial elements for learning processes and structures, which can—once identified—be negotiated. Lave and Wenger do not only focus on the structures and processes of learning in organizations but on its 'situatedness' and therefore on power structures which inform learning on the personal as well as the discursive level. Referring to their work, Bresnen, et al. (2005, p.39) found that networks of practice create their own logic of action. They can support or resist changes of routines, norms and values, and therefore, power structures. Consequently, the condition of communities of practice and their repertoire of actions must be considered, when it comes to the negotiation of power structures related to—for example diversity relations.

In order to highlight the connection between situated learning and power structures, Allhutter and Hofmann (2010) link the concept of communities of practice to the concept of 'trading zones' developed by Kellogg, et al. (2006). 'Trading zones' can be seen as real or virtual spaces of negotiation and learning or agreed procedures of exchange, which are more or less intentionally created by organizations and communities of practices. Such 'trading zones' are determined by power structures and must be identified when learning should be fostered in the cross-disciplinary teams commonly used in software engineering. The concept of 'trading zones' highlights how teams and communities of practice use certain spaces to coordinate actions and, also, to exchange and negotiate ideas, terms, norms, meanings, values and performance criteria (Kellogg et al., 2006, p.39). Another useful concept is the concept of 'boundary objects' introduced by Star and Griesemer (1989). It explains how e.g. communities of practice use objects, symbols or language for their cooperative activities. According to Star and Griesemer (1989, p.46), 'boundary objects' are objects that are both plastic enough to adapt to local needs and constraints of the several parties employing them, yet robust enough to maintain a common identity across sites. They are weakly structured in common use, and become strongly structured in individual-site use. Such objects may be quality standards, classification systems, databases, shared vocabulary, etc. Communities of practice or teams share various 'boundary objects' but their members may have different definitions of these objects.

In conducting collective work, people coming from different social worlds frequently have the experience of addressing an object that has a different meaning for each of them. Each social world has partial jurisdiction over the resources represented by that object, and mismatches caused by the overlap become problems of negotiation (Star & Griesemer, 1989, p.412). Both—'trading zones' and 'boundary objects'—are embedded in and informed by societal power structures and hegemonies.

4 'DECONSTRUCTIVE DESIGN'

This section outlines how the mentioned concepts can be integrated in a software engineering process. A design process is situated within specific contexts and (temporarily) affiliated team members bring different viewpoints to their common goal. Such communities of practice develop agency within a particular societal, economic and organisational framework. Furthermore, their socio-technological work practices are context-specific in that teams apply their established 'ways of doing' in particular projects. Their partly shared and partly diverging perspectives frame their boundary objects, i.e. concepts guiding the engineering process such as software quality standards.

Thus, as a first step the team identifies a boundary object crucial to its actual design process or latently present in its work practices. The second step is to investigate the different representations of the boundary object that team members have constructed for themselves and as a collective (e.g. different conceptions of quality). The different perspectives reflect structural positions of team members and socio-technologically constructed meanings (e.g. of intangible quality criteria), which implicitly inform work practices. Boundary objects implicitly serve as communication mechanisms that mostly are unreflected vehicles for commonly held beliefs. In heterogeneous communities of practice, they are essential for mutual knowledge creation and knowledge transfer, thus it is useful to investigate the different representations of the identified boundary object that team members have constructed for themselves. While diverging viewpoints may hinder common understanding and reaching goals, collectively shared constructions may also silence alternative ways of knowing. Thus, reflecting on individually and collectively shaped constructions of meaning will open negotiations and widen scopes of action. In order to disclose established practices and theories-in-use, 'mind scripting' is used to deconstruct collective processes of constructing meaning around the boundary object (e.g. anticipated quality requirements of particular user groups). Since common practices need to be revealed in a process of negotiating meaning within the group of designers, 'mind scripting' is organised as a collective procedure. It enables a team to research its cooperative work practices which are informed by their unconscious constructions of the boundary object. Deconstruction aims at temporarily disclosing an outsider's perspective to the team members and at enquiring the sense-making that permanently re-establishes what has implicitly been taken for granted. Deconstructing a boundary object crucial to situated specification and implementation practices, and investigating how societal discourses implicitly inform seemingly technology-centred concepts and decisions, enables the explicit negotiation of facets that are otherwise silenced. Furthermore, this process helps reveal and question structures, beliefs and value systems that reproduce dominant viewpoints. Basically, 'mind scripting' works with written texts representing memories that become operative in the actual design process. These texts are understood as narratives that developers use to give meaning to their experiences and practices. The collective deconstruction of the texts and the comparison of their sense-making processes disclose collectively shared meaning constructions.

In a third step, the meaning of the boundary object and its underlying theories-in-use are re-negotiated. After the 'mind scripting' engineers have revealed the consequences of unconsciously narrowing down the boundary object fundamental to their work practices. The opening of 'trading zones'—that means the establishment and explicit negotiation of 'boundary objects'—offers a way to integrate heterogeneous viewpoints and implicitly shared perspectives. This procedure creates space for negotiation and renders negotiable formerly unconscious issues. Initially blurred 'boundary objects' to which team members have imputed their views are made explicit; their meaning is re-negotiated and more clearly specified with regard to their value implications. While team members or subgroups of the team still approach the 'boundary object' from their specific perspective, their understanding of their own conceptions and of those of other members are both broadened and specified. Eventually, for all team members, the negotiated 'boundary object' still carries diverse but transparent meanings. In this way, they become a useful resource that enables better communication and diversity within teams.

5 CONCLUSIONS

A deconstructivist methodology to software engineering aims at reflecting collective work practices which unconsciously reproduce hegemonic discourses and unquestioned 'ways of doing'. It provides an analytic and interventionist approach to disclose the societal dimensions inherent to development practices. 'Deconstructive design' offers a procedure for development teams to open 'trading zones', i.e. to establish a space for negotiation and to

identify layers of 'boundary objects' that have been silenced and are worth negotiating consciously. Whereas 'mind scripting' has been used in two empirical case studies with commercial development teams, more empirical and practice-based research needs to be done to elaborate a practicable method-supported process implementing a sustainable learning routine. Whereas innovation in software engineering practice commonly tends to be narrowed down by unconsciously neglecting the implicit normativity of unreflected work practices, such a process shall widen scopes of action and incite cooperative process improvement. Analytical and socio-politically oriented STS research, as well as the value-related dimensions of critical technical practice add to SE research by connecting crucial societal aspects with process-oriented questions of practice-based feasibility and applicable methods.

REFERENCES

Akrich, M., 1995. User Representations: Practices, Methods and Sociology. In: A. Rip, T. J. Misa & J. Schot eds., 1995. *Managing Technology in Society. The Approach of Constructive Technology Assessment*. London, New York: Pinter Publishers, pp.167-84.

Argyris, C., 2002. Double loop learning, teaching, and research. *Academy of Management. Learning and Education,* 2(2), pp.206-18.

Allhutter, D., in review. Mind Scripting. A deconstructive method in software development. Submitted to *Science, Technology and Human Values*.

Allhutter, D. & Hanappi-Egger, E., 2005. Making the Invisible Visible: Mind-Scripting as Method of Deconstructing (IT-)System Design. In *Proceedings and CD-ROM of ICWES13*, KWSE.

Allhutter, D. & Hofmann, R., 2010. Deconstructive Design as an Approach to opening Trading Zones. In: J. Vallverdú ed., *Thinking Machines and the Philosophy of Computer Science: Concepts and Principles*. Hershey, PA: IGI Global, in print.

Bresnen, M., Goussevskaia, A. & Swan, J., 2005. Organizational Routines, Situated Learning and Processes of Change in Project-based Organizations. Project Management Journal, 3(3), pp.27-41.

Butler, J., 1990. *Gender trouble. Feminism and Subversion of Identity*. New York, London: Routledge.

Coleman, G. & O'Connor, R., 2007. Using grounded theory to understand software process improvement: A study of Irish software product companies. *Information and Software Technology,* 49(6), pp.654-67.

Dittrich, Y., John, M., Singer, J. & Tessem, B., 2007. Editorial for the special issue on Qualitative Software Engineering Research. *Information and Software Technology,* 49(6), pp.531-39.

Dittrich, Y., et al., 2008. Cooperative method development. Combining qualitative empirical research with method, technique and process improvement. *Empirical Software Engineering,* 13(3), pp.231-60.

Dittrich, Y., Randall, D. W. & Singer, J., 2009. Software Engineering as Cooperative Work. Editorial. *Computer Supported Cooperative Work,* 18(5/6), pp.393-99.

Flood, R. & Romm, N., 1996. *Diversity Management. Triple Loop Learning.* Chichester: J. Wiley.

Foucault, M., 1971. *L'archéologie du savoir.* Paris: Gallimard.

Friedman, B., Kahn, P. H. & Borning, A., 2006. Value Sensitive Design and information systems. In: P. Zhang & D. Galletta eds., *Human-computer interaction in management information systems: Foundation*. New York: AMIS, pp.348-72.

Hanappi-Egger, E., 2006. Gender and Software Engineering. In: E.M. Trauth ed., *Encyclopaedia of Gender and Information Technology*. Hershey, London: Idea Group Publishing. pp.453-59.

Hedberg, B., 1981. How organizations learn and unlearn. In: P.C. Nystrom & W.H. Starbuck eds., *Handbook of Organizational Design*, Vol. 1. New York: Oxford University Press, pp.3-27.

Kellogg, K. C., Orlikowski, W. J. & Yates, J., 2006. Life in the Trading Zone: Structuring Coordination Across Boundaries in Postbureaucratic Organizations. *Organization Science,* 17(1), pp.22-44.

Lave, J. & Wenger, E., 1991. *Situated Learning. Legitimate Peripheral Participation.* Cambridge, U.K.: Cambridge University Press.

McAvoy, J. & Butler, T., 2007. The impact of the Abilene Paradox on double-loop learning in an agile team. *Information and Software Technology,* 49(6), pp.552-63.

Mathiassen, L., 1998. *Reflective Systems Development*. Aalborg: Institute for Electronic Systems, Department of Computer Science. Available at: http://www.mathiassen.eci.gsu.edu/rsd.html [Accessed 5 August 2009].

McKenzie, D. & Wajcman, J., 1999. *The Social Shaping of Technology*. 2nd ed. Buckingham: Open University press.

Nonaka, I. & Takeuchi, H., 1995. *The Knowledge-Creating Company. How Japanese Companies Create the Dynamics of Innovation.* New York: Oxford University Press.

Oudshoorn, N. & Pinch, T. eds., 2003. *How Users Matter: The Co-Construction of Users and Technology.* Massachusetts: MIT Press.

Rip, A., Misa T. & Schot, J. eds., 1995. *Managing Technology in Society. The Approach of Constructive Technology Assessment.* London, New York: Pinter.

Robinson, H., Segal, J. & Sharp, H., 2007. Ethnographically-informed empirical studies of software practice. *Information and Software Technology,* 49(6), pp.540-51.

Rönkkö, K., 2007. Interpretation, interaction and reality construction in software engineering: An explanatory model. *Information and Software Technology,* 49(6), pp.682-93.

Sengers, P., Boehner, K., David, S. & Kaye, J., 2005. Reflective Design. In *Proceedings of the 4th Decennial Conference on Critical Computing: Between Sense and Sensibility.* ACM Press, pp.49-58.

Star, S. L. & Griesemer, J. R,. 1989. Institutional Ecology, 'Translations' and Boundary Objects: Amateurs and Professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39. *Social Studies of Science*, 19(4), pp.387–420.

Suchman, L., 2007. *Human-Machine Reconfigurations. Plans and Situated Action*. 2nd ed. Cambridge: Cambridge University Press.

Trauth, E. M., 2001. *Qualitative Research in IS: Issues and Trends*. Hershey: IGI Global.