

Researching The Social Side of Software Engineering

Yvonne Dittrich

The social side of Software Engineering is still an under-researched area. Existing publications about qualitative social science methods and approaches are inadequate. This article gives an overview of the state-of-the-art in this developing field, identifies the central challenges and highlights ways to address them; how to make software development visible, how to combine the methods borrowed from Social Sciences with software process and method improvement, and how to handle the political side of this type of action research.

Keywords: Cooperative Work, Qualitative Software Engineering, Social Sciences, Software Process.

1 Introduction

"No rule is defining what it means to follow it." [39] And, in the same sense, no method defines what it means to apply it. So how do software engineers make sense of methods and apply them in concrete circumstances? How do they coordinate this deployment? How do they make process models work, how do they use planning and coordination tools? These questions address Software Engineering as a cooperative work practice, as a social endeavour.

The social side of software development has been identified from the very beginning as one of the key success factors [23][24][9]. Indeed, in the special track on Software Engineering for the next millennium, social science inspired research on the social aspects of Software Engineering was addressed as important future line of research [13]. Nevertheless, relatively little research is performed within the Software Engineering community and less is published in central journals or presented at the important conferences held so far. The social side of Software Engineering and qualitative research seems to take place mainly in workshops. [19][34]

The reason for this might be that the social science approaches which aim at understanding how people make methods, organisations and procedures work are mainly the qualitative ones. Also, qualitative methods are notoriously bad at producing isolated causal relationships that can easily be translated into software process improvement measures.

Qualitative methods allow the development of an understanding of the social side of software development from a members' point of view. [29] By addressing Software Engineering as a social achievement, these methods provide rich descriptions of how software engineers make the software and its development work. Such research not only helps us to understand whether methods work or do not work, but also to understand how and why they work (see e.g. [11]). This understanding can then be used to improve methods and tools.

This article gives an overview of the existing research addressing the social aspects of Software Engineering published across different scientific discourses. Thereafter, the main problems of transferring social science methods into Software Engineering are summarised; the specific charac-

ter of software as an invisible, highly malleable, and complex product [5]; the fact that Software Engineering, as a discipline, aims at improving the development practice; and that therefore Software Engineering researchers are influencing their research subject in another way than the major part of the Social Sciences. Here we propose ways to address these difficulties. The article thus gives an overview of the state of the art and the difficulties of researching the social side of Software Engineering and points at ways to address these difficulties

2 An Overview over Existing Research

As mentioned in the introduction, research on the social aspects of Software Engineering is distributed over different discourses. Roughly, one can distinguish the research published in the mainstream of Software Engineering, the research published as computer supported cooperative work, and a number of researchers often from Scandinavia publishing mainly in the information systems discourse. Of course these discourses partly overlap.

The intention of this article is not to give a complete overview of existing research but to give interested researchers and practitioners' pointers to start further investigation of the research discourses presented here.

2.1 Qualitative Research in The Software Engineering discourse

One of the earliest and most widely cited qualitative studies on Software Engineering is 'A field study of the software design process for large systems' from 1988 where communication and coordination is indicated as the most crucial factors for software development [9]. The article cites research from the mid 70s and 80s both from the empirical

Author

Yvonne Dittrich works as Associate Professor at the IT-University of Copenhagen, Denmark. Dr. Dittrich's research interests are use-oriented design and development of software, end especially the flexibilisation of software products and processes in order to accommodate the co-development of work practices and technology. She developed an empirical research approach Cooperative Method Development together with industrial partners which is based on problem-oriented software process improvement as a learning cycle both for the industrial partner as well as for the researchers involved. <ydi@itu.dk>.

research of programmers and the engineering management communities. Qualitative research has been part of the empirical research on Software Engineering from the very beginning. Nonetheless, it has been quantitative research that dominated and dominates the empirical research in Software Engineering.

Researchers addressing the empirical side of Software Engineering have so far mainly applied quantitative research methods.

Basili, with his work on experimentation in Software Engineering and the Software Engineering Laboratory at the University of Maryland, laid the ground for this approach [1][2]. Software engineering research should develop models of and methods for software development and analyze and evaluate them.

One of a recent example of this kind of research is an article [18] on an empirical theory of coordination in Software Engineering.

However, not all aspects of communication and cooperation are easy to measure. Regarding this aspect, Seaman proposed to complement quantitative methods with qualitative ones. [32]

She regards qualitative research methods mainly as a complement in early research stages, a way to generate hypotheses that then can be tested quantitatively.

As the goal is to understand the mechanics of Software Engineering, i.e. the influence of the deployment of certain methods on the outcome of Software Engineering, the goal often is to identify quantifiable causal relationships. Seaman reports from qualitative observations of inspection meetings that are coded in a quantifiable way so that statistics can be applied.

Other research published in Software Engineering venues triangulates qualitative results with quantitative methods. One of the means for quality assurance of qualitative empirical research is the deployment of different methods and data sources or the cooperation of different researchers during the analysis to counter possible individual or methodological biases. [35] provides an example of this kind of triangulation, results from questionnaires, participatory observations, interviews and tool usage statistics were combined in order to understand work practices of software maintenance and compile a list of requirements for a software exploration tool.

Relatively few publications present purely qualitative research. [19] reports a study of coordination in distributed development based on interview data and document analysis [11]. reports how a traditional process model was used as a frame for iterative implementation and user-developer cooperation. [33] reports an ethnographic study of XP (Extreme Programming) practice as a culture that is based on and supported by the XP practices but reaches beyond 'implementing' them. All these studies have in common that they focus on understanding of how software engineers make the development work, rather than identify dependent and independent variables respectively relationships between them.

2.2 Software Engineering as Computer Supported Cooperative Work

As software development is a co-operative effort it has become subject of discussion in Computer Supported Cooperative Work (CSCW) [31]. Traditionally empirical research in CSCW mainly uses qualitative methods: Ethnography, ethnomethodologically informed ethnography, for example, often combines with participatory design processes as action research.

Issues are; the use of representations and design work as embodied practice [37], organizational constraints and their influence on the work practice [6][7], the development of organizational patterns from within the project group [27][28][38] or the interaction of work practice and computer based tools [14][15]. The research focus here is on understanding the ways the members of a Software Engineering project achieve coordination of their cooperative effort. A more recent example of this kind of research is [30] where the authors explore how software engineers used plans to coordinate a widely distributed method and tool development project.

This kind of research is mainly rooted in ethnography. The researcher tries to understand software development as work practice from within. This sometimes leads to what, for software engineers, looks like the appreciation of a skillfully performed bad practice. The interesting question – what makes a disadvantageous practice less troublesome than changing the habit? – is seldom asked. Few authors seldom use the studies to further develop methods or tools for developers. Here the work of Grinter [14] provides an exception.

2.3 'Out of Scandinavia'

In reference to the seminal introduction of what non-Scandinavians call the Scandinavian Schools of Systems Development to the international research community [40] this section presents a regionally rooted strand of research, which is, internationally, mainly published in information systems venues. As the Scandinavian School on Systems Development is characterised by a humble attitude towards the expertise of the future users of the software under development, this strand of research takes the experience of the practitioners as a starting point [3][4][21][12].

Research is performed as Action Research meaning that researchers take part in industry in software development or software process improvement. The active participation is complemented by qualitative and quantitative data collection, and a detailed documentation of their own intervention.

This whole data provides the basis of the research focusing on the evaluation of the introduced measures. In [22] a major project on software process improvement is reported which involved researchers from different Danish universities and a number of industrial partners. This project resulted in a number of publications as well as addressing the social and organizational conditions for software processes and their improvements (e.g. [25]).

Due to the connection to the Scandinavian approach to systems development, the relation between user participation and Software Engineering methods and processes is a continuous thread of discussion within this community.

Already [3] discusses the adequacy of methods to support user developer communication. [28] addresses the constraints that software methods and processes put on developers regarding the possibility of taking usability into account. [16] reports on a case where agility in the development processes facilitates quick reaction to customer feedback, even in product development. [11] explores how the use of oriented software development can take place. In 2004 a special issue of the Scandinavian Journal of Information Systems addressed *inter alia* the implications of a change in the relationship between use and development of web-based systems. [8]

3 The Challenges of Qualitative Research

As discussed above, qualitative research, on the one hand, provides a better understanding the social of software development. On the other hand it faces special difficulties that have to be addressed when undertaking qualitative empirical research. Already quantitative research has to struggle with the problem that it is not easy to define simple measures of software as a product beyond lines of code or function points.

For observational methods the fact that software is invisible provides even more of a challenge. Doing research within an engineering discipline requires the focus to be not only on understanding what takes place, but also on generating improvements based on this understanding. This requires an extension of qualitative research into action research. And last but not least, doing qualitative fieldwork the researcher gets more involved with the social and political context in which Software Engineering takes place. This has to be reflected in both the design of the empirical work and the analysis of the results.

3.1 Making Software Engineering Visible

Software is inherently invisible, and software engineers themselves work with different sets of representation, models, and specifications of which the source code of the final product is one [5]. This poses problems for all observational fieldwork methods. Observational methods rely on the coordination of activity that partly takes place through the joint artefact being observable in the manipulation of the artefact. This, in Software Engineering, is only partly possible, e.g. when observing design discussions in front of a whiteboard, as e.g. in [37].

However, this will only allow the observation of a small part of the communication mediated through the common artefact. Another possibility is to study the coordination of work through a common repository, a configuration management system e.g. But here the observation of the concrete action that leads to the changes in the files is difficult. Examples of this approach to making the complexity and cooperation of distributed work visible can be found in [18].

We ourselves experimented with techniques to map out the development process in cooperation with the involved practitioners [11]. Here the project members themselves took part in the task to represent the complex network of activities, documents and people involved in the development process. Here the map itself provides only part of the field material, the other extremely important part is the taping and transcription of the discussion which takes place when constructing the map with the involved practitioner. It also requires constructing a medium or notation for the representation that is geared towards the research question at hand. And it should be triangulated with other observational and interview methods, to counter possible biases by the involved practitioners, respectively to allow understanding the background for the specific way the members present their reality.

3.2 From Understanding to Improvement

Qualitative social science methods are geared to make understandable the social aspects from a member's point of view. This insight allows an understanding of how people (and in our case software engineers) manage to handle the cooperation and communication to achieve their task.

The effort involved in understanding the complexities of this achievement, e.g. the interlace of project models, planning tools, social arrangements like role specific responsibility and meeting patterns for the coordination of a distributed development project [30], sometimes keeps the researcher from readily proposing tools or other means to help the problem's practitioners in their daily practice. And it keeps researchers from stepping in the 'bad practice' trap that attracts many engineering researchers: "*If the practitioners just would have used the tools and methods the right way, this would solve their problems*" (see also [29]). The understanding of how methods, processes and tools actually inform the practice of software engineers is in itself an important contribution. However, as an engineering discipline, Software Engineering is not only interested in understanding how practitioners cooperate around the development of software, but also how this process can be supported with method, processes and tools.

Relating to the Scandinavian tradition of action research in participatory design we developed a research approach that designs and implements the improvement in an evolutionary cycle of empirical research, design of improvement measures together with the practitioners involved, and evaluation of the implemented improvements again with empirical means [10][12]. In this way we retain the strength of the qualitative empirical approach – understanding the social practice of software development from within – even through the improvement and method invention/adaptation process. Our experience so far supports the use of Cooperate Method Development frame for research cooperation with industry.

3.3 Handling The Political Side of Research

Even experimental or quantitative Software Engineer-

ing research might be subject to social dynamics and politics within the field of enquiry as e.g. the failing trial to introduce elements of cleanroom software development in industrial practice reported in [2] shows. When doing qualitative research, the researchers gets involved with the practice they observe in a way that lets the ethical and political aspects of the research and the improvement proposals become more exposed.

Already the very act of seeing and representing how Software Engineering actually takes place can have a political dimension, e.g. when the research shows how developers circumvent mandatory procedures (see [36] for a more general discussion.)

The researchers' positioning in relation to the political and power hierarchies within the organisation in which the research takes place will influence what the researchers get to see and what kind of improvements are possible (see [20] as an example for such influence and how it can be dealt with).

Being explicit about the researchers' perspective, the ethical handling of the empirical data – e.g. rigorously implementing the member checking procedures agreed on – and being explicit about base and rational for the improvement proposals is therefore mandatory for empirical research in industrial practice.

4 Conclusions

Though qualitative research addressing the social side of Software Engineering is not yet visible as an identifiable strand of the Software Engineering discourse, we can identify a rich variety of research approaches and results. One major problem for this kind of research is that the publications are distributed over different discourses.

Few individual researchers are publishing in more than one discourse. References across the different discourses are infrequent. Researchers that publish in more than one community can serve as brokers between the different communities.

Qualitative research on Software Engineering is not easier than quantitative research. It provides a different set of challenges when implementing the research, both in terms of data collection, analysis, and argumentation. Qualitative research is not better than quantitative research. It addresses a different set of questions and provides complementing results. These complementing results can contribute to a richer understanding of how methods are used and why they work or don't work. Taken together, the existing qualitative research on Software Engineering provides a sound base for mature research.

References

- [1] V. Basili. The role of experimentation in Software Engineering: past, current, and future. Proceedings of the ICSE '96, pp. 442-449, 1996..
- [2] V. Basili, S. Green. Software Process Evolution at the SEL. IEEE Software, pp. 58-66, 1994.
- [3] J.P. Bansler, K. Bødker. A Reappraisal of Structured Analysis: Design in an Organizational Context. ACM Transaction on Information Systems, 11 (2), pp. 165-193, 1993.
- [4] J. Bansler, E. Havn. The nature of Software Work. Systems Development as Labour Process. In: Pvd. Besselaar et al.: Information System Work and Organisation Design. Elsevier Science Publication, pp.145-153, 1991.
- [5] F.P. Brooks Jr. No Silver Bullets – Essence and Accidents of Software Engineering. Computer, pp. 1027-1037, 1987.
- [6] G. Button, W. Sharrock. Occasional practice in the work of software engineers. In: Jirotko, M., and Goguen, J. Requirements Engineering. Social and Technical Issues. London, 1994.
- [7] G. Button, W. Sharrock. Project Work: The Organization of Collaborative Design and Development in Software Engineering. Computer Supported Cooperative Work, Special Issue on Studies of Cooperative Design, 5 (4), pp. 368-386, 1996.
- [8] K. Bødker, P. Carstensen. Development and Use of Web-Based Information Systems. Scandinavian Journal of Information Systems 16, pp. 3-10, 2004.
- [9] B. Curtis, H. Krasner, N. Iscoe. A field study of the software design process for large systems. Communications of the ACM 31, pp. 1268-1287, 1998.
- [10] Y. Dittrich. Doing Empirical Research in Software Engineering – finding a path between understanding, intervention and method development. In: Y. Dittrich, C. Floyd, R. Klischewski (eds.). Social thinking – Software practice. MIT Press, 243-262, 2002.
- [11] Y. Dittrich, O. Lindeberg. How use-orientated development can take place. Information and Software Technology 46, pp. 603-617, 2004.
- [12] Y. Dittrich, K. Rönkkö, O. Lindeberg, J. Eriksson, C. Hansson. Cooperative Method Development revisited. Human and Social Factors of Software Engineering (Workshop at the International Conference on Software Engineering 2005).
- [13] A. Fuggetta. Software Process: A Roadmap. In: Finkelstein, A. (ed.) The Future of Software Engineering, New York: ACM, pp. 25-34, 2000.
- [14] R.E. Grinter. Supporting Articulation Work using Software Configuration Management Systems Computer Supported Cooperative Work, Special Issue on Studies of Cooperative Design, 5 (4), pp. 447-465, 1996.
- [15] R.E. Grinter. Recomposition: Coordinating a Web of Software Dependencies. Computer Supported Cooperative Work, 12, pp. 297-327, 2003.
- [16] C. Hansson, Y. Dittrich, D. Randall. Agile Processes Enhancing User Participation for Small Providers of Off-the-Shelf Software. In: Extreme Programming and Agile Processes in Software Engineering. Proceedings of the 5th International Conference, XP 2004, Garmisch-Partenkirchen, Germany, June 6-10, 2004.
- [17] J.D. Herbsleb, R.E. Grinter. Splitting the Organization and integrating the Code: Conway's Law Revisited. 21st

- 21st International Conference on Software Engineering, Los Angeles, CA, pp. 85-95, 1999.
- [18] J.D. Herbsleb, A. Mockus. Formulation and Preliminary Test of an Empirical Theory of Coordination in Software Engineering. In: proceedings, European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering, Helsinki, Finland, September 1-5, pp. 112-121, 2003.
- [19] HSSE. Human and Social Factors of Software Engineering, held in conjunction with 27th International Conference on Software Engineering (ICSE 2005), 16 May, St. Louis, Missouri, USA, 2005.
- [20] J.K. Iversen, P.A. Nielsen, J. Nørbjerg. Problem Diagnosis Software Process Improvement. In: T. J. Larsen, L. Levine, J.I. DeGross (eds.) Information systems: Current Issues and Future Changes IFIP 1998.
- [21] L. Mathiassen. Reflective Systems Development. *Scandinavian Journal of Information Systems*, 10(1&2), pp. 67-118, 1998.
- [22] L. Mathiassen, J. Pries Heje, O. Ngwenyama. *Improving Software Organizations*. Addison-Wesley, 2002.
- [23] P. Naur. *Computing: A Human Activity*. Addison-Wesley, 1991.
- [24] K. Nygaard. Program Development as a Social Activity. In: H.J. Kugler (eds.) *Information Processing 86*. IFIP, pp. 189-198, 1986.
- [25] P.A. Nielsen, J. Nørbjerg. Assessing Software Processes: Low Maturity or Sensible Practice. *Scandinavian Journal of Information Systems*, 13, pp. 23-36, 2001.
- [26] J. Nørbjerg, P. Kraft. Software Practice is Social Practice. In: Y. Dittrich, C. Floyd, and R. Klischewski (eds.) *Social thinking – Software practice*. MIT Press, pp. 205-212, 2002.
- [27] C. Potts, L. Catledge. Collaborative Conceptual Design: A Large Software Project Case Study. *Computer Supported Cooperative Work, Special Issue on Studies of Cooperative Design*, 5 (4), pp. 415-445, 1996.
- [28] T. Robertson. Embodied Action in Time and Place: The Cooperative Design of a Multimedia. *Educational Computer Game Computer Supported Cooperative Work, Special Issue on Studies of Cooperative Design*, 5 (4), pp. 341-367, 1996.
- [29] K. Rönkkö, Y. Dittrich, and O. Lindeberg. "Bad Practice" or "Bad Methods" – Are Software Engineering and Ethno-graphic Discourses Incompatible? *Proceedings 1st International Symposium on Empirical Software Engineering (ISESE'02)*, Nara, Japan, pp. 204-210, 2002.
- [30] K. Rönkkö, Y. Dittrich, D. Randall. When Plans do not Work Out: How Plans are Used in Software Development Projects, accepted at the *Journal of Computer Supported Cooperative Work*, 14 (5), pp. 433-468, 2005.
- [31] K. Schmidt, W. Sharrock. *Computer Supported Cooperative Work, Special Issue on Studies of Cooperative Design*, 5 (4), pp.369-386, 1996.
- [32] H. Sharp, H. Robinson. An Ethnographic Study of XP Practices. *Empirical Software Engineering*, 9, pp. 353-375, 2004.
- [33] C. Seaman. Qualitative Methods in Empirical Studies of Software Engineering, In *IEEE Transactions on Software Engineering*, 25(4), pp. 557-572, 1999.
- [34] S. Sim, J. Singer, M. Storey. Beg, Borrow, or Steal: Using Multidisciplinary Approaches in Empirical Software Engineering Research, *Proceedings of Empirical Software Engineering*, 6(1), pp. 85-93, 2001.
- [35] J. Singer, T. Lethbridge, N. Vinson, N. Anquetil. An examination of Software Engineering work practices. *Proc. CASCON. IBM Toronto*, 209Y223, October, 1997.
- [36] L. Suchman. Making Work Visible. *Communications of the ACM*, 38 (9), 1995.
- [37] L. Luchman, R. Trigg. Artificial Intelligence as craftwork. In: Chaiklin, S., Lave, J. *Understanding Practices – Perspectives on Activity and Context*. Cambridge University Press, NY, pp. 144-178, 1993.
- [38] H. Tellioglu, I. Wagner. Negotiating Boundaries. *Configuration Management in Software Development Teams. Computer Supported Cooperative Work*, 6, pp. 251-274, 1997.
- [39] L. Wittgenstein. *Philosophical Investigations*, Basil Blackwell& Mott, Ltd., 1958.
- [40] C. Floyd, W.M. Mehl, F.-M. Reisin, G. Schmidt, G. Wolf. Out of Scandinavia: Alternative Software Design and Development in Scandinavia. *Journal for human Computer Interaction*, 4, pp. 253-380, 1989.