# Beg, Borrow, and Steal – But What, and What For?

**Yvonne Dittrich**
Department of Software Engineering and Computer Science
University of Karlskrona Ronneby
Softcenter
S37225 Ronneby, Sweden
+46 457 385842
yvonne.dittrich@ipd.hk-r.se

**Abstract:**

Software engineering has a fundamentally different agenda than most social science approaches. It aims at the change and improvement of software development practice. The choice of social science methods and their necessary adaptation depend very much on how we conceptualise this change process and how we view the role of our industrial partners in that process. Taking a recently started research project as an example, I argue for a co-operation between software engineering practitioners and researchers regarding possible improvements. Social science methods then provide – as in Participatory Design of computer applications – important input for the co-operative development of methods.

## Introduction

Software Engineering is – beside other things – a social activity. [10] For sure, social science methods will help us to understand more about it. So the whole spectrum of social science research from ethnography via action research to quantitative analysis and measurements offers itself as possible research approaches. How to decide which ones to pick? Should you as a software engineering researcher join one school and bring the old arguments from the humanities over to software engineering? And won't you become a social scientist doing research about a growing subculture? Beside such 'political' questions there are a whole bunch of epistemological issues popping up: How to counteract your own bias when using qualitative methods? Isn't a quantitative set-up from the very beginning excluding everything beside the parameters you want to measure? And are they relevant for the question we pose? And last not least: How should we adapt these methods, that are meant to come to an understanding of social processes to our discipline that has a declared goal to improve software development?

## Changing Focus

Instead of examining the different methods, compare them, their scope and so on, I propose to reconsider the reason to beg, borrow, or steal from social sciences. Why is empirical research considered of growing importance during the last years? One answer can be found in Robert Glass' ranting against software engineering research as ivory towerish and arrogantly advocating its mindchilds upon practice. [5] And true, software engineering was founded rather to invent a practice than to fulfil a growing need for abstractions, concepts, and models necessary in the further development of an existing and successful practice. (See all the reference to the software crisis, or the historical summaries around the 25th anniversary of the Garmisch-Patenkirchen Conference, for example [3].) Another might be the recognition that software engineering practice is more and more diversifying. Software development takes place under different economical, organisational and cultural conditions. The development of concepts, methods and tools has to take into consideration the situational context where they will be used in order to provide usable products.

So it is actually a problematic relation to software development practice that seems to stand behind the growing interest in understanding software development as a social activity. So why not 'take the bull by its horns' and reconsider our relation to the practice we try to inform?

**How to relate to practitioners?**

Even as software engineering as a research discipline was founded in a programmatic way, to design a practice, practice was taken serious in form of experience reports providing important feed back. During the last years, however, a more reflected relation to industrial practice developed: Two in some respects quite opposite approaches are the experimental approach to software engineering by Basili et al. [1, 2] and the reflective systems development approach by Mathiassen [8, 9].

Basili proposes in [1] that software engineering should follow the model of other physical sciences and develops an experimental paradigm. Software engineering research should develop models and methods about software development and analyse and evaluate them. Beside thought experiments software engineering researchers need laboratories to test their hypotheses. As realistic settings are important, research and industry should build up such laboratories together. Researchers can develop and evaluate models that fit to the specific company and help to improve the software development their. Industrial practice provides a realistic set of projects for experimentation.
How such research might look like is described in [2]. As an example Basili and Green report several experiments with key elements of the cleanroom development method; replacing testing by reading and peer review. In the first set of experiments graduate students and personal at the Software Engineering Laboratory – a co-operation between the University of Maryland and the NASA Goddard Space Flight Center – used reading and testing on programs seeded with errors. The different approaches were quantitatively compared. In a second step, lab groups of a university course were partly assigned to use cleanroom development; partly they used traditional testing methods. Here again the different development approaches were evaluated regarding their performance in error detection. With positive results from both experiments, two case studies in NASA projects were designed and evaluated. Emphasis is put on the measurable variables making the improvement visible in a quantitative manner. The co-operation between researchers and practitioners seems to be more a co-operation between researchers and software development management. The observées are observed, their performance is measured, analysed and evaluated. However, they do not seem to participate in the set-up of the experiments and the case studies, and they do not seem to participate in the design of the improvement and the evaluation criteria.

Mathiassen from Åalborg University in Denmark describes his research approach to software engineering in [8, p 80f]: 'The problems, challenges and opportunities involved in systems development practice are considered the starting point for systems development research. Research activities yield experience-based knowledge that leads to new, and hopefully improved practices. The knowledge that is developed is both interpretive, helping us to understand and make sense of practice, and normative, providing support for performing systems development or improving present practices.' Systems development practice is understood according to Schön's description of the 'Reflective practitioner. [11] The practitioner is active in research as he makes his thinking and his strategies visible for the researchers and uses the researchers result in a support to his own reflection-in-action. '[T]he reflective researcher…', Mathiassen cites Schön, '… cannot maintain distance from much less superiority to, the experience of practice … he must somehow gain an inside view of the experience of practice.' [8, p 81] Research is performed as Action Research, meaning that researchers take part in software development projects, software process improvement projects and so on in industry. To understand practice several forms representations are introduced that, used by the practitioners themselves, provide also material for the research. [9, chapters 2, 3, 4] Formalisations regarding methods and processes are understood as such helpful representations as well. [9, chapter 5] Action research is complemented by case studies and experiments where necessary and suitable. [8, p 81]

The approach I propose here can be regarded as a variation of the latter. It shares the high regard for the practitioners competence. In a way it perhaps radicalises the approach by inviting software developers to take part in the method development process. Concepts, methods, process models, project organisation and planning tools as well as case tools can be understood as tools for software developers. As in the design of computer applications, one can claim, that domain experts, that means practitioners have a better understanding of what is suitable to support them in their everyday work practice. A difficulty might be that software engineers develop a 'professional blindness' that means they perform their tasks without themselves recognising the details of action and communication it takes to get the work done. During the last years ethnographic studies and other methods are used to complement Participatory Design by providing additional input.

[7] The field studies inform the co-operative design of methods, models and tools as well as the evaluation of these meetings and an observation and evaluation of the application of the designed measures provide a base for generalisation and reflection. A similar approach in a software process improvement project is reported in [6]. In the following I will exemplify this approach outlining a research project just started:

## The Project: Design in Use of Database Applications

The project tries to explore the deployment of a database server that allows for the change of the data model during use [4]. The scenario that is subject to the project under discussion is the support for rapidly developing business areas. The project is a co-operation between three parties; a small software development company providing the technology, a growing telecommunication provider in southern Sweden and a group of researchers at our university department. Telecommunication is a developing and rapidly changing business, providing suitable application domains for this kind of technology. One of their projects provides the case study hopefully helping us to answer our research questions. Beside concrete design issues regarding design, functionality and user interfaces that are not in focus here, methodological and organisational questions are to be solved: How far can users change an application? When does a change become a task for maintenance and further development? How is the distribution of work between users, technical support and software engineers effected? How is the co-operation between the three parties organised? Should not already the development be organised as an evolutionary process, getting as much early feed back as possible? How to accompany the users through the beginning phase that can be expected to be shaky? How to document and keep track of the changed implemented by the users? With respect to the growing deployment of information technology in developing domains, like for example the use of internet technology for the integration of several data sources for developing public services, the results can be expected to be useful in other contexts, too.

In the first phase the database server is completed with a data model manipulation client and a first simple generic client, so its applicability for the considered domain can be estimated. The case study projects starts by performing a prestudy. Parallel we conduct a reconstruction of the history of the application that should be replaced. This reconstruction should make visible the dynamics of change and help us to understand the software development culture, the methods and models and their application at the IT department we are co-operating with. Depending on the results of this phase, there will be a decision on what should be implemented and how.

## Methodological considerations

To answer the above questions regarding the changes in the organisation of the developing process, we plan to adapt a set of methods actually developed for the design of computer applications. Not only CASE tools but also concepts; process models, project organisation methods and documents can be regarded as tools for software developers and software development groups. Our experience regarding the design of socially embedded software, that it does not answer the needs of the users, seems to apply for our own tools too. This fall we plan to start with a series of workshops together with the involved developers in order to identify relevant issues and together design suitable process models and ways of co-operation between users, technical support and developers. These means will be evaluated and refined together with the project members throughout the project. Parallel to the continuation of the workshops we plan to observe the project and the implementation of the designed measures in a participatory way. Again the field material should provide on one hand input for the co-operative workshops and, on the other hand, a sound base for scientific evaluation of the improvements. We plan to use qualitative methods from ethnography. However we are open for any kind of method providing helpful input to the co-operative development of methods. For example we plan to do experiments and measurements comparing change effort using traditional technology vs. deploying the new flexible database technology.

## Autobiographical Background

Yvonne Dittrich works as assistant professor in the area of software engineering and heads the research group 'Use Oriented Design and Development of Software'. Already during my undergraduate studies I began to also read courses in other disciplines, like pedagogic and data privacy law. During my PhD I went deeper into philosophy of technology and language philosophy and also used qualitative case studies. The resulting thesis 'Computer and language context – About the interaction between formal and ordinary language during development and use of computer applications' uses philosophical reflection as well as empirical studies to answer research questions from within computer science. At the University of Karlskrona/Ronneby, I contribute to the study program 'People, Computer, and Work' combining human work science and computer science and developed together with colleagues an interdisciplinary project course 'Work Practice, Design and Development of Software'. Two other members of the same research group, Olle Lindeberg and Kari Rönkkö, have sent in a position paper as well.

## References

1. Basili, V. The role of experimentation in software engineering: past, current, and future. *Proceedings of the ICSE '96*, pp442-449.
2. Basili, V. Greeen, S. Software Process Evolution at the SEL. *IEEE Software* July 1994: 58-66.
3. Bauer, F.L. Software Engineering – wie es begann *Informatik Spektrum* (1993) 16:259-260.
4. Diestelkamp, W. Promis, a Generic Product Information Database System. Lee, in: R. Y. (Ed.) *Proceedings of the ISCA14th International Conference*, Cancun Mexiko, April 7-9, 1999.
5. Glass, R. L. The software Research Crisis *IEEE Software* November 1994: 42-47.
6. Iversen, J. K., Nielsen, P. A., Nœrbjerg, J. Problem Diagnosis Software Process Improvement. In: T. J. Larsen, L. Levine, J.I. DeGross (eds.) *Information systems: Current Issues and Future Changes* IFIP 1998.
7. Kensing, F., Simonsen, J., Bødker, K., MUST: A Method for Participatory Design. *Human-Computer Interaction,* 13(2) 1998, 167-198.
8. Mathiassen, L. Reflective Systems Development. *Scandinavian Journal of Information Systems* 1998, 10(1&2):67-118.
9. Mathiassen, L., *Reflective Systems Development.* Dr. Tech. Thesis, Aalborg University, 1997.
10. Nygaard, K. Program Development as a Social Activity. in: Kugler, H.J. (Eds.) *Information Processing 86.* IFIP 1986, 189-198.
11. Schön, D. A. The Reflective Practitioner. How Professionals Think in Action. Basic Books 1983.