Some Thoughts about the Horizontal Development of Software Engineers

Anke Dittmar and Peter Forbrig

Rostock University, 18055 Rostock, Germany
{anke.dittmar,peter.forbrig}@uni-rostock.de

Abstract. We argue that current patterns of thought and action in software engineering and in HCI will simply be reproduced if we are not able to become more aware of their impact on our own behaviour, attitudes and values. We suggest that a more balanced and intertwined vertical and horizontal development of people can contribute to human-centred design processes. The case study presented describes a modest attempt to demonstrate this with future software engineers and managers. Though not a spectacular example, it shows a small tight network of activities and roles over time with feedback loops to facilitate deep reflection, mutual awareness and respect. The paper supports the idea of design as an ongoing intervention process beyond problem setting and problem solving.

1 Introduction

Diaper points out in [1] that "HCI is most closely related to the computing field of software engineering" and that "no distinction should ever have been made between software engineering and HCI because both are engineering disciplines concerned with the same types of systems and their difference is merely one of emphasis, with software engineering focusing more on software and HCI more on people." However, "integration of software engineering and user-centred design" is the first topic mentioned in the call for papers of this conference. Obviously, there is still a gap between the two approaches. The title HCSE even goes a step further by suggesting not to focus on *users* of technology but on *humans*.

Why is software engineering not inherently human-centred? One explanation is that there is always a lag between the invention of new technologies and the learning of how to use them in a 'reasonable' way. This is also reflected in HCI. According to Cockton, its focus has expanded from being largely system-centred up to the 1970s, then user-centred in the 1980s, context-centred in the 1990s, and now, having a value-centred focus [2]. Appropriate design approaches have been developed since then to improve the design of interactive systems (task-based design, participatory design, design rationale, reflective design, end user development,...). On the other hand, the engineering side of system development is often underestimated today (as stated e.g. in [3]). While ten or twenty years ago users and other stakeholders were often seen as not able to contribute to the design process (and this view might still prevail in software engineering) the HCI field tends to consider now interaction programmers as

P. Forbrig and F. Paternò (Eds.): HCSE/TAMODIA 2008, LNCS 5247, pp. 213–220, 2008.

mere executors of other people's ideas. Maybe this is a kind of counter effect. However, it also shows how difficult it is to really accept contributions from different fields and different people. It is one thing to understand the rationale behind a new approach. It is another thing to internalize those ideas and to bring them into balance with existing habits of thought and action. To give another example, we still struggle to find a balance between so-called formal, semi-formal, and informal approaches and representations, and we are often not even aware that they can share similar assumptions which, perhaps, should be questioned first.

What does it mean to do human-centred software engineering, or maybe value-centred design or sustainable design? Why, for example, does Thimbleby give at the beginning of his book about principles of interaction programming [3] explanations about interests behind short production cycles, about toxic waste, time pressure on programmers and its consequences? Blevis states in [4] that the material effects of current practices of designing and using interactive systems do not reflect a sustainable lifestyle. He proposes several design principles to increase our understanding of the environmental impact of interaction design. However, he also suggests "that faith in technology as usual cannot succeed, and that new thinking is critical to our survival." All this is not new. Einstein is frequently cited (e.g. in [5]): "The world we have created is a product of our thinking, it cannot be changed without changing our thinking".

In this paper, we briefly describe a series of tutorials in requirements engineering with graduate students of software engineering and business informatics (Sec. 2). We will refer to it as a case study though it was not planned as such. However, its specific conditions and its evolution triggered the reflective analysis, and the suggestions for learning practices, which are presented in Sec. 3. We argue that it is not enough to reflect on external products of design activities. Current patterns of thought and action will simply be reproduced if we are not able to become more aware of the impact of actual practices on our own behaviour, on our attitudes and values. We suggest that deep reflection and a more balanced and intertwined vertical and horizontal development of people can contribute to more effective human-centred design processes. Though we do not ground this work in a sound qualitative methodology but rather remain on a descriptive level we think that our reflective analysis can contribute to a more sustainable software engineering culture in which design is understood as an ongoing intervention process beyond problem setting and solving.

2 Case Study

The case study is about tutorials supplementing the requirements engineering lectures (RE) at Rostock in summer 2007. Participants were 15 graduate students of software engineering and business informatics. They were familiar with programming, formal specifications and with software engineering methods. They also had done an internship. The focus of such tutorials is on the early stage in a user-centred design process. Their nature is partly shaped by the following constraints. The participation is optional. The main interest of most students is neither in RE nor in HCI. Even in their industrial training, many had no experience of any deep requirements analysis. Students get no marks or points and their participation is not a prerequisite for other

courses or examinations. Hence, the focus of a tutorial is more on the activities in the 11-12 weekly meetings (each about 90 minutes) and less on the production of precise specification documents (though documents are produced). We do not insist on training in particular methods by using specific, independent examples. Instead, a single 'problem' is used throughout the semester. Some of the methods and techniques which were introduced in lectures are chosen to approach the problem. They are mostly applied in a sketchy way. In addition, the meetings are used to reflect personal activities, to see improvements and alternative approaches, and to discuss pros and cons of the artifacts in use.

The reported case study was about analysing the software engineering course (SE) for second-year students at our department in order to find out how to better support student projects. Both authors are involved in the SE course as well. We could ask our colleagues and students to 'act' as participants. We also chose this topic because of the obviously different, and partly conflicting, views of the stakeholders. The following description is based on material created during the tutorials and on the notes of the tutor. Sometimes the first person is used to emphasize that it is the perspective of the tutor (one author).

First Meeting

The students were asked to work in groups and develop initial ideas of how to tackle the analysis. Most participants started by reflecting their own past experiences with the SE course. Some students discussed, for example, whether the goal of a project is to learn about object-oriented software development or to work in a team. One of them said: *I am sure, if you went to one of the teachers right now to ask them about the goals of these projects they would make up a story.* The whole group agreed on conducting semi-structured interviews with the teaching staff involved and with second-year students. Two students were asked to prepare a test interview with the tutor (who also was SE tutor). Eleven students were asked to make appointments with the teachers.

Interviews

The test interview in the second meeting might have helped to make sense of the grading scheme or to understand better the work of tutors. However, it was also obvious that the questions had to be revised to get more insights. The revised list guided the interviews of the other four tutors and the lecturer. Generally, all interviews were recorded. They were transcribed (with differences in detail). We used Stud.IP (a learning management system with wiki support at our university) to store and access audio files and all other documents. In the third meeting, we listened to a 40-minute interview with one of the tutors. In the subsequent meetings only transcriptions were used. We did not perform a thorough analysis. Instead, we rather used the interviews for a kind of 'informed dialogue' between ourselves about the SE course and about project work in particular.

The tutors were asked to help us contacting one or two of their project groups. In another meeting we divided into three groups to prepare a list of questions for students. One group wrote down 'ad hoc' questions. Another group was asked to develop a simple task model from the student's perspective before writing down their questions. The third group created a list of artifacts used in the SE course and developed questions on this basis. Then, all questions were gathered, selected, and grouped.

Finally, four interviews with at least three students of a group (and two interviewers) were conducted. In one case, the whole group was present. The interviews took from 40 to 70 minutes. At that time, I suggested that we should try to organize a 'workshop' at the end of the RE course.

Brainstorming
The following list shows an extract from suggested improvements at the 'brainstorming session' in the 7th meeting. Based on this list, we planned the last four meetings. The group decided to invite teachers but not students for a final workshop "SE 2.0". All invited people were present.

1. Registration: choice of project topic, group formation,
2. More relations between documents of a project,
3. Management tool for teaching staff,
4. Announcement of the SE course, e.g. invitation of the first-year students to the final project presentations,
5. More milestones in the second phase of the project (summer semester),
6. More exchange between project groups, e.g. mutual testing, code reviewing...

Specifying Requirements
In the 8th meeting, we began to explicitly describe requirements. We used different techniques such as use cases and paper prototypes. Prepared material facilitated the meetings. For example, an entity-relationship diagram encouraged consideration of flexible graduation schemes, and helped to find requirements on a management system for tutors. Fig. 1 shows the revised version of the "rough QOC" developed during the discussion about future registration practices for student projects.

3 A Reflective Analysis

Development is often understood as 'vertical' improvement of individuals though supported by social interaction and collaboration [6]. However, vertical development also needs a horizontal movement across social worlds. The T-model (e.g. [7]) is well-known but not necessarily practised in education. The | can stand for the vertical and the ¯ for the horizontal development. Engeström mentions 'contact zones' as places where people and ideas from different cultures meet, collide and merge. "It is this inability to ever understand another world that has great developmental significance" [6]. Participatory design supports this idea. Authors like Schön initiated a transformation of design by promoting a "reflective practice" with argumentation processes and an intertwined goal shaping and problem solving [8].

Sec. 2 may have already shown that the way we organize the tutorials is rooted in these ideas. They can be seen as complementing 'classrooms' and project work. Students are neither evaluated nor forced to create a 'visible product'. The idea is to support a horizontal development but with a 'starting point' which is familiar to the participants (requirements engineering is part of software development). There are few pre-defined roles and goals. There is no pre-established agenda. The idea is to create a continuous conversation about current and envisioned practices in a certain

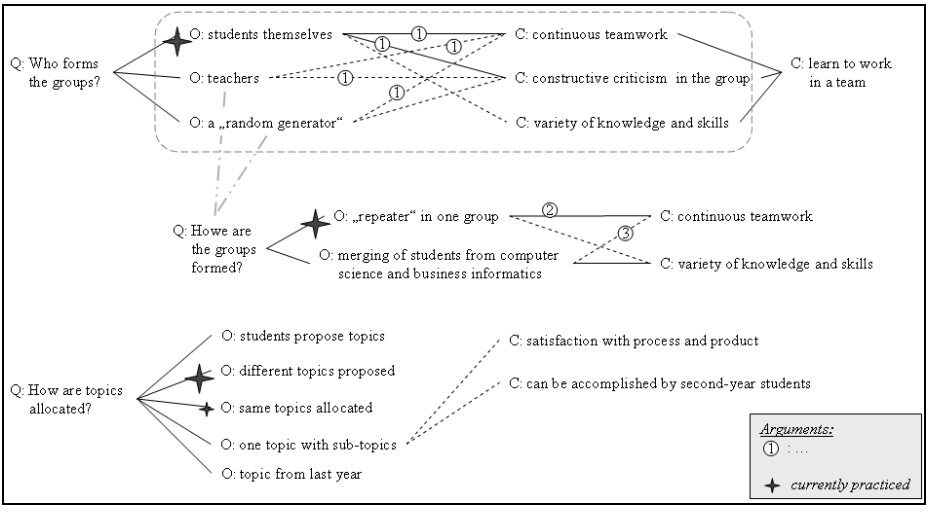

Fig. 1. QOC diagram (8th meeting)

working system. A conversation which is guided by early design techniques, most of them well-known or even developed in the HCI field. The focus is on a continuous experience and less on the creation of 'perfect' artifacts. Mistakes are allowed. It is also allowed to use suggested techniques in a sketchy way. Perhaps this facilitates a combination of child-like playfulness and adult-like rationality as recommended in [9].

The case study might be used to illustrate several points. For example, none of the students was trained in conducting interviews. Of course, we made mistakes. The first tutor showed some surprise when students started to record the interview. We never forgot again to ask and to emphasize that we don't want to 'test' the interviewees. Suggestive questions were asked. One interviewer asked a tutor: *Do you REALLY read the documents of student's project?* What can he answer? As mentioned, interviews were not thoroughly analyzed. However, it is possible to hand out and discuss a description like the following (it only takes five minutes). "The interviews were recorded and transcribed. Analysis included open coding for thematic analysis, selective coding and constant comparison between analysis products and raw data...". Students can recognise themselves in the description but also see that much more knowledge, experience, and work(!) is required. Perhaps, this helps to create a deeper understanding of other stakeholder's activities, an appreciation of diverse viewpoints, and mutual respect.

In the specific study described in Sec. 2, the authors were responsible for the RE course. Participants were graduate students in software engineering and business informatics. They analysed a basic course in software engineering with activities of second-year students, the authors, and other tutors. Most of them attended this basic course some semesters ago. Hence, students were teachers and teachers were learners in a way. Who were the users and who the developers, who the observers and who the observed? As it turned out, a 'frame' was set up which was convenient to support.

- Multiple, sometimes blurred roles and actions with multiple motives,
- Intertwined vertical and horizontal development,
- Deep reflection,
- Mutual respect and shared understanding,
- The idea of design as ongoing intervention.

Vertical and Horizontal Development
SE projects are basically guided by the waterfall system life cycle. In a way, the older students 'observed' their own activity of two or three years ago but now through the lenses of their increased knowledge and skills in software engineering. Some of the suggested early design techniques are not yet applied in the 'real world' of software development. They require an understanding and skills which are often not conveyed in 'traditional' software engineering. The case study might be a modest example of an intertwined vertical and horizontal development. The ¯ in the above mentioned T-model is deeply related to the | and yet different ways of thinking and acting are needed.

Deep Reflection
In [10], ongoing and off-loop reflection is required for a professional participatory design process. Off-loop reflection is seen as an opportunity to reify and discuss past experiences, and to establish a firm link to possible future practices. As already described, both forms of reflections were evoked. Two small examples from the 8th meeting about new registration practices for projects may serve for illustration. One group of participants applied use cases [11], the other used the concept of "rough QOC" [12]. Then, we looked at the notes of both groups to compose a proposal. The 'nature' of the approaches literally emerged. The use case with its focus on action sequences looked like a 'first-best solution' in comparison with the QOC diagram. Though the three questions in Fig. 1 seem to be trivial they don't have simple answers, let alone a best one. However, look at Fig. 1 again to understand the following situation in the QOC discussion (written from the tutor's perspective who 'served' as QOC scribe to record the discussion): *One student said that there are fewer conflicts and more continuous work if students can form their own groups. They know each other, their skills and so on. So, I drew a solid line between the appropriate option and the criterion. Then, another student said that he is not sure about that argument. It could also be a handicap to be friends and work in a team. I changed to a pencil and drew a dashed line between the same option and the same criterion. After two more arguments I drew a big question mark over this part of the paper. A student asked: Are we allowed to do it?! I said: Of course. This is a sheet of paper and we write and draw what we want to. This led us to a 10 minutes 'philosophical' talk about modeling, programming, the need for intertwining different activities in software design and so on. One student described, for example, some of his problems with modeling. There is a term for that: premature commitment to structure.* This situation may reveal much about how we teach and live with cognitive artifacts like methods. Is it allowed to draw a circle in a diagram which normally consists of rectangles and arrows? Or, is it allowed to perform step 4 of a method before step 2? And doing this without rejecting the whole method? It looks like a paradox. On the one hand, there is often rejection of methods or rules, on the other hand, a kind of faith in them.

Mutual Respect and Shared Understanding
We think one reason why the students became engaged in the analysis was that they were not detached observers. For example, the interviews were sometimes more like an exchange of experience and knowledge. One interviewer explained to a group who didn't use a version management system what it is good for. Of course, the participants were more experienced and had better understanding than the second-year students. However, they were still students and saw us as teaching staff though in a more relaxed way. We think there was much potential in this tension. The group started to see their own assumptions and was sometimes a kind of mediator. This might be illustrated by the following interview situation: *The students started to complain about a tutor (not theirs). The interviewer said he notices it but it has no consequences for anyone. The students said that they would like to let us know about it. The interviewer said: "Okay, this analysis is about improving the SE course. And teachers who are not committed will be chained to the wall and whipped. You are all invited to come." Laughter and the interview could continue.*

Design as Ongoing Intervention
Even in a "reflective design practice" with an intertwined goal shaping and problem solving the problem is still the main underlying concept - whether wicked or tame. In contrast, the idea of design as an ongoing process of a double intervention "in the Earth's cycles and processes, and simultaneously in the human culture of needs and techniques" [13] may be better supported by Bohm's idea of embedding problem solving into *awareness of paradoxes*. What is called for in the case of a paradox is not some procedure that solves the problem. Rather, it is to pause and to give attention to it in order "to bring the root of the paradox into awareness" [14]. Bohm suggests that the treatment of paradoxes as problems and the attempt to solve them does not contribute to their dissolving but results in "ever-increasing confusion". In the case study presented there are paradoxes between education and practice, between the desire of students to get good (individual) marks and yet to learn teamwork, between methods and actual situations... Take note that the brainstorming session was in the second half of the tutorial. We think that the relatively long first phase was an important experience for the participants. It helped to suspend activities of problem solving and to become aware of paradoxes. Perhaps this resulted in more 'modest' suggestions for changes at the workshop. Some of them are considered in the actual SE course, some of them were the basis for actual SE project topics.

4 Summary

"[T]hrough centuries of habit and conditioning, our prevailing tendency is now to suppose that 'basically we ourselves are all right' and that our difficulties generally have outward causes, which can be treated as problems" [14]. The paper is not about another method 'to bridge the gap' between SE and HCI. It looks instead for ways to facilitate a cooperative internalisation of non-familiar ideas and perspectives in order to question and change one's own practices. The case study presented describes a modest attempt to demonstrate this with future software engineers and managers. Though not a spectacular study it is a small example of a relatively tight network of activities and roles over time with feedback loops supporting deep reflection, mutual

awareness and respect (including self-awareness and self-respect). We are not able to validate our suggestions but we would like to encourage others to look for 'seeds' for adaptations in their own (design) attitudes and activities. Human-centred software engineering has to be treated as paradox, not as problem. There are no answers in terms of solutions (or methods). Design concepts and methods like those mentioned in this paper are artifacts that can guide this process. However, they cannot free humans from the need to be aware of the actual situation and the need to adapt it in a sensitive way. This includes the questioning and revision of the very same artifacts.

References

1. Diaper, D.: Understanding Task Analysis for Human-Computer Interaction. In: Diaper, D., Stanton, N.A. (eds.) The handbook of task analysis for human-computer interaction. Lawrence Erlbaum, Mahwah (2004)
2. Cockton, G.: A Development Framework for Value-Centred Design. In: Proc. CHI 2005. ACM Press, New York (2005)
3. Thimbleby, H.: Press On: Principles of interaction programming. MIT Press, Cambridge (2007)
4. Blevis, E.: Sustainable Interaction Design: Invention & Disposal, Renewal & Reuse. In: Proc. CHI 2007. ACM Press, New York (2007)
5. Ackoff, R.L.: Transforming the Systems Movement. In: ICSTM 2004 (2004), http://www.acasa.upenn.edu/RLAConfPaper.pdf
6. Engeström, Y.: Development as Breaking Away and Opening Up: A Challenge to Vygotsky and Piaget. Swiss Journal of Psychology 55, 126–132 (1996)
7. Dix, A.: Controversy and Provocation (Keynote). In: Proceedings of HCIE 2004, The 7th Educators Workshop: Effective Teaching and Training in HCI (2004)
8. Schön, D.A.: The reflective practitioner: how professionals think in action. Harper Collins (1983)
9. Dix, A.: Being playful: learning from children. In: Proc. IDC 2003: Interaction Design and Children. ACM Press, New York (2003)
10. Bødker, S., Iversen, O.: Staging a Professional Participatory Design Practice - Moving PD beyond the Initial Fascination of User Involvement. In: Proc. NordiCHI (2002)
11. Cockburn, A.: Writing Effective Use Cases. Addison-Wesley, Reading (2001)
12. Buckingham Shum, S., MacLean, A., Bellotti, V., Hammond, N.: Graphical Argumentation and Design Cognition. Human-Computer Interaction 12(3) (1997)
13. Knapp, R.: Sustainable Design, http://diac.cpsr.org/cgi-bin/diac02/pattern.cgi/public?pattern_id=798
14. Bohm, D.: On Dialogue. Routledge (1996)