Between Cooperative Creativity and Conflicts on Appreciation: Customer-developer-links in small software companies

Andrea Sieber
Chemnitz University of Technology
Department of Computer Science
Chemnitz, D 09107, Germany
+49 371 531 1388
andrea.sieber@informatik.tu-chemnitz.de

ABSTRACT
Research on participatory design focuses on scientific projects and model applications; few studies draw on commercial settings. In such settings the customer and its wishes are seen as dominant. The paper explores the nature of customer-developer-links in commercial projects. How can the cooperation between customers and developers be described? Based on case studies made in previous projects, I show that customer-developer-links oscillate between inspiring cooperation and fights for power and appreciation. Is there a chance to improve and professionalize these links? Answering this question needs further research.

Keywords
Participatory design, customer-developer-links, design practice, small enterprises

1. INTRODUCTION
Participatory Design is a concept with several assumptions. In its origin it is related to the Scandinavian context of software development. It implies a certain development frame and refers to a set of values in software development such as a democratic involvement of all relevant user groups and social integration of technology. Although the context and conditions of user involvement are specific to the Scandinavian countries and their strong trade union traditions, approaches to user involvement are not. Research groups in other countries developed with or without reference to the Scandinavian origin their own approaches, for instance JAD (Joint Application Design) in the US (Carmel, Whitaker, and George 1993), the ETHICS (Effective Technical and Human Implementation of Computerbased Systems) Approach in UK (Mumford 1993), STEPS (Software Technology for Evolutionary Participatory Systemdesign) in Germany (Floyd et al. 1989).

User participation has its most references in research projects (Clement, Van den Besselaar 1993). It is originally connected with action research, but also experiments and studies of development practice (Mathiassen 1998). There are some reports that examine commercial software development and implementation and their suitability for participation (e. g. Greenbaum, Stuedahl 2000; Törpel 2000, McCormack, Forlizzi 2000). In general it is understood that participative approaches are not widely used in enterprises. And that is seen as a problem (Kensing 2000).

According to my research there are much more commercial projects that could be described as participative, user centered and moreover user led – if software developers would have known participative approaches, and would have had a broader understanding of participation that includes commercial development contexts.

Keil and Carmel (1995) have already introduced a broader view to participation. They identify a whole list of customer-developer-links. Concerning their terms: 'customers' include all people somehow infected by the software system; 'developers' are all individuals directly involved in the design and production of code. Links between both groups are "techniques and/or channels that allow customers and developers to exchange information" (p. 33). Some of these links have already been described substantially at participative design approaches, others not. They altogether build a rich base to identify participation in different forms in commercial contexts.

This broader view to participation is an argument to take a closer look into customer-developer-links in the every day work life. Since I have been dealing with the working practices of software developers in small enterprises in different regions and countries in two empirical projects[1] it seems worth checking the available cases before drawing a new research project. This paper presents the findings of this pre-study with main emphasis on situations in which developers report on forms of cooperation with customers.

The following section describes the empirical data and how it has been collected and analyzed. After a classification of the companies in the sample I give short descriptions of reported situations crucial for the customer-developer-relationship. The findings will be discussed and a project idea is sketched out in the last section.

In PDC-04 Proceedings of the Participatory Design Conference, Vol 2, Toronto, Canada, July 27-31, 2004, under a Creative Commons license. CPSR, P.O. Box 717, Palo Alto, CA 94302. http://www.cpsr.org ISBN 0-9667818-3-X

[1] One project was funded by the German Research Foundation (DFG) from 1999 to 2002 (grant No. Bo 929/13-1). The other project is funded by the Volkswagen-Foundation from 2003 to 2005 (grant No.II/78383).

2. DATA SOURCE

Case studies of small software companies had been conducted in a project starting in 1999. Six companies were located in the western part of Germany, five in the eastern part. The project aimed to analyze cultural differences in the software development process arising from these specific regional contexts. Due to history and economic conditions in both parts of Germany the most striking differences could be identified in the companies' networks, with firms from western Germany drawing on a wider net of contacts. Also, the companies differed in the range and technical quality of their product depending on their starting point (Henninger, Sieber 2001, also see below).

The collection of the empirical data in the other project has not yet been finished. Until now case studies have been conducted completely in the USA and partly in Germany. Nine companies were located in the Research Triangle (NC). Eight companies were situated in the Seattle Area (WA). Seven companies were analyzed in Southwest Saxony (Germany). In this project innovative development processes in relation to inner- and outer-organizational networking of the company are in focus. First results have been published in Henninger 2003.

Although the project settings differ the case studies have a lot in common. They are conducted in small companies because nearly 40 % of these IT-companies in the US (Hofmann 2001) and nearly 60% in Germany (Schwarz 2000) have just up to ten employees. The cases are analyzed based on theme-oriented interviews with the CEO and - after a field observation phase - with developers, and other core staff. The company's and especially the developers' everyday working practices are observed.

All interviews with CEOs included questions on the origin and development of the company, business areas, work organization and goals. All interviews with developers and key staff members asked for their professional biography, tasks, working day, working practices, self organization of work and goals. Through the observation notes architectural aspects of the location, working atmosphere, and emotions of the researcher came into play and were partially used as critical incidents in the interviews or for specific discussion to show the informal organization of work. These topics play a role in each case – besides others that are project specific. The material from the interviews and from the observations was analyzed through certain steps to interpretation according to Strauss and Corbin (1990).

3. CUSTOMER AND DEVELOPER

Although the empirical data were analyzed to identify differences between the companies according to the varying frames of organizational, regional and national settings, there were certain similarities. The first subsection presents some of these general characteristics with emphasis on the role of the customers. In the second subsection the developers' stories of relationships to customers are summarized and interpreted.

3.1 Characteristics of the Companies

Main differences in the organization of the software development process, the quality of the software and the networking of the companies could be traced back to the starting point of the company. Some companies had been started to develop a self-made software product, others to offer (IT-) services to customers. These different goals in the beginning were connected to a different path, organization and network of the company.

3.1.1 "Product first"-companies: the virtual customer becomes real

These companies that were found to produce and sell a self-made software product had developed a more (Germany) or less (USA) elaborated prototype. Intense prototype development had been done in Germany at the university. In the US a fragmental prototype had been developed besides the regular job. The founders of the companies had a technical educational background, in the US in addition experiences in other IT-businesses: research divisions of large companies or previous product development phases in other small and medium sized companies. By founding an own company the owners tried to raise money and time to develop a beta version of the product and to bring it to the market.

Financial sources for this state of development were in this sample: (1) state funded "technology to market" programs (eastern part of Germany), (2) industrial research projects (eastern part of Germany), (3) IT-service projects (western part of Germany, US), (4) venture capital (US), (5) private savings (US).

To produce a beta version of the product it was necessary to thoroughly redesign and complete the prototype. The expenses were way above the developers' estimations. It included additional and often disliked tasks as documentation, installation of a help and security system, interface-construction for "normal" users.

When the CEOs started talking to a first potential customer, the CEOs and the developers reported on an additional redesign: the customer-led redesign phase. In some companies this customer-led redesign happened parallel to the beta version development and caused a chaotic situation in the development department with lots of confusions, contradicting work tasks, and a lot of time pressure. That happened especially in the companies financed by venture capital.

Up to that point the picture of the end-users' requirements was more or less virtual. The first real potential customer gave the management and the software developers a more realistic view of the working conditions, the organizational restrictions, and the requirements of the end-users. The more realistic view of the end-users' work evolved through discussions with the customer, and visits at the workplaces of the end-users. They were combined with observation and interviewing of the end-users by management and developers.

The customer-led redesign phase caused again major reconstructions of the software – this time based on the beta version - in both software technical and usability dimensions. Metaphorically speaking it was necessary to develop packages of code that allowed a fast reconstruction of the software to meet the requirements of this customer. To reconstruct the code in such a way was a question of the technical software structure. In the usability dimension especially the layout of the interface, names, the hierarchy of functions, the speed of specific routines, the memory gathering of specific operations had to be changed and optimized. If this redesign was conducted thoroughly the software was flexible enough to handle the individual requirements of new customers and general technological changes in a very short time.

None of the observed companies could keep a software version for long, but there were two ways of changing the system. 1) It could be upgraded in a certain timeframe. This upgrade contained the most urging changes asked for by the end-users and was delivered to all of them. 2) Each customer got his individual assembled software system. Most companies followed 2), one was organized around the development of a yearly upgrade and two companies sent upgrades to all customers in case of general technological changes. For both ways it can be stated: The better the software was constructed technically the faster they could realize the adoptions produced by new or changed customer requirements or technological changes.

3.1.2 "Service first"-companies: software becomes a product

These companies started to establish their own income by selling IT- or other services. The companies offering IT-services in the beginning were selling computer systems and software. Through these services they found customers and increasingly concentrated in the adoption of a special "off-the-shelf" software product requested by their customers. The customers started to trust in their technical abilities and also asked them to develop software solutions for their specific needs. The founders of these IT-service companies had a technical background and had gained practical experience as technical employees or contractors. Nearly all of them at least dreamed of a self-made product while doing routine software development for "old" technology. They planned the development of the product out of the already existing code.

There were also companies with a self-made software product in the sample that had been starting with services in other branches as for instance consulting. The product idea arose with the increasing understanding of their customers' work. The CEOs realized the lack of appropriate software systems in their customers' application domain. Because the market for technical businesses was good until 2000 they found it worth a try to develop a self-made software system based on their knowledge about the customers. Because of their already established relationships to potential end-users of the system they had no doubt on their market success – if the system was well developed. Their lack of experience in software development supported a theoretical start by studying literature concerning: How to organize software development well? Based on their theoretical knowledge they hired one or several software developers and let them start.

Once the decision to develop an own software product was made by these "service first"-companies they had to go through all phases described in the paragraph before: prototype development, development of the beta version, customer-led redesign phase. The already established relationships to customers helped them to get comments on their work in every phase and to find buyers. They had to gain their own experience in how to organize the software development process and to construct software in such a way that it could easily be adapted to the changing end-users' requirements. To see the specific organizational and technical needs related to software product development was the greatest challenge to these companies.

Surprisingly the founders without technical background met at least the organizational challenges related to software development. Their experience at the customers' domain let them see the necessity to involve customers in the systems development. Their lack of technical knowledge and experience forced them to study software engineering literature, and figure out the pitfalls in a project's course. For the technical challenge they depended on the abilities and experiences of the hired software developers. If they were capable to see especially the need for organizing and structuring the code from a technical point of view, the development moved on.

The companies that started with offering IT-services were often not aware of both challenges of the software development process, as their software already functioned in the defined setting. However, if there were any – sometimes just slight – changes, this could cause a complete rebuilt. This lack of appropriate technical structure was due to the restricted time and budget of the usual projects. This lack had to be resolved in the software code used for the product. They had also often hired developers that were fond of programming but didn't see the need for structuring code appropriately. But a software that should work as a marketable product demands thoroughly structural rework. The CEOs with technical knowledge in general but without specific software development experience were susceptible to oversee this.

3.2 Relationships

In both types of software companies, relationships to customers evolved. Because of the smallness of the companies, and the fact that they tried to sell software systems, most of the software developers came in contact to customers and end-users. In the interviews the software developers described their relations to customers. Based on examples of such descriptions I show chances and pitfalls of this cooperation, arguing that in a best case scenario customer-developer-links can be fruitful for both sides, resulting in mutual support. On the other hand, potential problems of such cooperation include fights for authority and appreciation.

3.2.1 Examples from the interviews

Albrecht worked as a software developer to finance his studies in computer science. In his job he had to work with an end-user that was an expert in the application domain and fond of programming. Albrecht had to support him in questions of programming and to offer him software parts developed by the company. Albrecht was upset about the bad programming style of that customer. In addition he had the feeling, that the customer always accused him if there was an error in the code. He tended to postpone the customer's requests and avoided to contact him. Instead he changed the parts of the software that were programmed by the customer without talking to him.

Bert was a founder of a company and worked there as a developer. He described the work with two end-users as contrasting. One customer was really helpful in his opinion. The customer used his software parts, tried them out, told his preferences and discussed his ideas with the developer and vice versa. Bert had the feeling that this customer really appreciated his work and did never question his abilities. There were times they often called each other and times without any contact. But they could revitalize the relationship whenever it was necessary.

The other end-user Bert had to work with did never test the system in one step. He tested a part and if there was an error he immediately reported it to him and waited for repair before he tested again. Bert had the feeling that the customer was just looking for errors and for something to complain about to have a

reason to postpone the paying. Because of the customer's portioned testing he could never rework the system as a whole. Instead he had always to search for small error – a part of his work he hated. Bert postponed the error repair as long as possible and avoided to contact the customer.

Rudolf was also a founder of a company and worked as developer. He had to work with an end-user that always called for help. As he could only do serious programming without disturbance he postponed his own work to support this customer. Through his questions Rudolf got an understanding how end-users might see the system, about what they were complaining. From that knowledge he draw some consequences how to rework his software. The customer also gave him feedback on his programming ideas. Rudolf talked to the customer when he ran across him or when he got a call from him.

3.2.2 Interpretation

As the descriptions of the developers show, relationships to customers are sometimes characterized by fights for power and appreciation, sometimes by reciprocal motivation and empowerment. In the cases where the developers feel frustrated by their contact to the customer, they reflected on the relationship – maybe to be able to continue the work with the customer at all. They all knew that they depend on the customers and that they have to keep them satisfied. But they also could not ignore there feelings of anger and frustration.

Although the developers talk to each other about the customers there is no reported try to change frustrating relations: for example to talk to the customer on this subject or to give the work to a colleague who might be better able to deal with him. This might be interpreted as a lack of professionalism.

The descriptions also give the impression, that customers have different roles. There seem to be key customers that give insights to their field of expertise to the developers and discuss ideas with them. The relation to a customer can also change over time. There are periods of intense cooperation and times with no contact at all. If a relation is positive for both sides, that seems to be no problem. If not, misunderstandings and accusations may follow. There also seems to be a good chance that contacts to customers fade away as a side effect of postponed support and delayed responses.

4. CONCLUSION

Analyzing customer-developer-links in an extended view reveals the social embeddedness of such relations as stated by Granovetter (1985). Personal relations on friendly terms back up constructive cooperation enhancing software usability and innovative product ideas. Relations dominated by distrust and frustration are susceptible to conflicts that result in cooperation and paying problems. The handling of these problems seems not to be professionalized in the small companies in my sample. There seem to be different categories of customers. >From the perspective of the developers, the cooperative or conflicting nature of the relationship between customer and developer seems to arise by chance. Their interventions are restricted to influencing the intensity of the contact but not the basic character of the relationship.

However, in the projects' case studies I did not focus on the relationships between customers and developers. The example descriptions are biased on the developers view and there are other elements that play a role in these links, as stated by Greenbaum and Stuedahl (2000). They analyzed the web design and development process and found on base of the Actor-Network theory time as a relevant (nonhuman) actor. In addition they identified several boundary objects that "were shared and shaped among the professional groups" (p. 71). A further research has to take these elements into account.

The general differences between small software companies support the differentiation between several forms of development made by Keil and Carmel (1995) and partly by Grudin (1991). Keil and Carmel differentiate between custom and package development. Grudin (1991) identifies three types of development settings: product development, contract development and in house development. In my sample the software development process of the last two types did not differ, except that the last one was mostly realized at the customer's location.

Each setting – "product first" or "service first" – seems to have its own advantages and disadvantages. On the one hand the "product first"-companies got a thoroughly structured software system and could meet the needs of varying end-users or changing technological requirements very fast. On the other hand they had a hard time to find enough customers to survive economically. The "service first" companies developed usable software that was on the contrary very expensive to change. The already existing network to customers didn't prevent them from the expenses necessary to improve the beta version in terms of usability and especially technical structure. To oversee or underestimate this last redesign step could result in a failing market implementation of the system.

It seems worth while to explore customer-developer-links further. What are the preconditions of a successful cooperation? How can developers influence the quality of the relationship? Which roles play nonhuman actors like time and boundary objects as for example contracts and prototypes? To answer these questions a thoroughly designed research project has to be conducted in which both sides – customers and developers – have to be observed and talked to. The results could lighten the already existing working practices, support reflection and increase professionalism on that part of the process. The examples in the interviews already show: Customer-developer-links as fruitful cooperation of both sides are a valuable resource for the development process, a fundamental part to the existence of the company and to innovative software systems.

5. ACKNOWLEDGMENTS

Thanks for valuable comments on earlier drafts of this paper go especially to Annette Henninger, Werner Dilger, and Mandy Kräuter.

6. REFERENCES

Carmel, E., Whitaker, R. D., and George, J. F. PD and Joint Application Design: A Transatlantic Comparison. In Communications of the ACM, 36, 4 (June 1993), 40-49.

Clement, A., and Van den Besselaar, P. A. Retrospecitve Look at PD Projects. In Communications of the ACM, 36, 4 (June 1993), 29-37.

Floyd, C., Reisin, F.-M., Schmidt, G. STEPS to Software Development with Users. In Ghezzi, C., McDermid, J. A.

(Eds.) ESEC '89, Lecture Notes in Computer Science No. 387, Springer, Berlin, 1989, 48-64.

Granovetter, M. Economic Action and Social Structure: The Problem of Embeddedness. American Journal of Sociology, 91, 3 (Sept. 1985), 481-510.

Greenbaum, J., Stuedahl, D. Deadlines and Work Practices in New Media Development: Its about time. In Proceedings of the PDC 2000. CPSR Press, Palo Alto, CA, 2000, 70-77.

Grudin, J. The Development of Interactice Systems: Bridging the gaps between developers and users. IEEE Computer, 24, 4 (April 1991), 59-61.

Henninger, A., Sieber, A. Softwaredevelopment in Small Companies in the Eastern and Western Part of Germany. In Matuschek, I., Henninger, A., Kleemann, F. (Eds.) Neue Medien im Arbeitsalltag: Empirische Befunde-Gestaltungskonzepte-Theoretische Perspektiven. Westdeutscher Verlag, Opladen, 37-54.

Hofmann, L. Research Triangle Park. In Triangle City Facts: The triangles business almanac, Signatur, Raleigh, 2001, 34-41.

Keil, M., Carmel, E. Customer-Developer-Links in Software Development. In Communications of the ACM, 38, 5 (May 1995), 33-45.

Kensing, F. Participatory Design in a Commercial Context: A conceptual framework. In Proceedings of the PDC 2000. CPSR Press, Palo Alto, CA, 2000, 116-126.

Mathiassen, L. Reflective Systems Development. Scandinavian Journal of Information Systems, 1, 1& 2 (June 1998), 67-118.

McCormack, M., Forlizzi, J. Listening to User Experience: Integrating technology with proactive wellness management. In Proceedings of the PDC 2000. CPSR Press, Palo Alto, CA, 2000, 296-300.

Mumford, E. The ETHICS Approach. In Communications of the ACM, 36, 4 (June 1993),82.

Schwarz, A. Diverging Patterns of Informalization between Endogenous and Exogenous Exonomic Actors in the East German Transformation Proess: Results from a case study in the IT-Branch in Berlin-Brandenburg. Working Paper, Frankfurter Institut für Transformationsstudien, Frankfurt (Oder), Nov. 2000.

Strauss, A., Corbin, J. Basics of Qualitative Research: Grounded theory procedures and techniques. Sage, Newbury Park, 1990.

Törpel, B. Self-employed Labor meets Codetermination – Participatory Desgin in Network Organizations. In Proceedings of the PDC 2000. CPSR Press, Palo Alto, CA, 2000, 184-191.