2007

Improving Information Systems by End User Development: A Case Study

Christian Dorner
University of Siegen, christian.doerner@uni-siegen.de

Jan Heß
University of Siegen, jan.hess@uni-siegen.de

Volkmar Pipek
Universität Siegen, volkmar.pipek@uni-siegen.de

IMPROVING INFORMATION SYSTEMS BY END USER DEVELOPMENT: A CASE STUDY

Christian Dörner, University of Siegen, Hölderlinstraße 3, 57068 Siegen, Germany, christian.doerner@uni-siegen.de

Jan Heß, University of Siegen, Hölderlinstraße 3, 57068 Siegen, Germany, jan.hess@uni-siegen.de

Volkmar Pipek, University of Siegen, Hölderlinstraße 3, 57068 Siegen, Germany, volkmar.pipek@uni-siegen.de

Abstract

This paper presents a case study in which we identified problems end users have with their software systems and how they try to solve these problems. The case study is part of a bigger research project, in which we develop appropriate end user development tools for enterprise resource planning systems of small and medium sized enterprises. During our study, we carried out several interviews with employees of such enterprises from different industries. The results allow a deep insight into the dynamics of adapting software to the users' application domains and are valuable for improving the quality of information systems by making them more adaptable to users' requirements. The aim of developing appropriate information systems for small and medium sized enterprises could be achieved by end user development techniques, allowing end users with little technical knowledge to adapt their systems during use time. Furthermore we make suggestions how end user development techniques could be instrumented for this purpose.

Keywords: *case study, end user development, qualitative research, software adaptation, IS design.*

1 INTRODUCTION

Despite of its long research tradition, the construction of software systems which exactly meet the needs of all users' in every situation is still a challenge for software designers. (Suchman, 1987; MacLean et al., 1990; Stevens et al., 2005). As a result, software systems often do not fully correspond to the current working context. Reasons for this divergence are the changing needs of users and the frequently changing economic requirements (von Hippel & Katz, 2002), which are affected by the dynamic markets in which these companies have to compete. To survive this competition, the companies' software systems have to be adaptable to organizational requirements, which are also influenced by the markets.

Over the last decade many researchers tried to figure out how software systems could be made more flexible and how end users could participate in the design and construction of software systems to overcome these issues. The outcome was a wide range of possibilities, including recent approaches such as end user computing (Nardi, 1993), tailoring (Stiemerling et al., 1997), meta-design (Fischer et al., 2004) and end user development (*EUD*) (Lieberman et al., 2005b). Fischer (2006) states that collaborative design, social creativity and meta-design are important software engineering topics for the years to come. Liebermann (2005a) assumes that the future goal of software engineers will be not "just making systems easy to use" but "making systems that are easy to develop", meaning end users should be able to adapt their systems. The end user development approach is the best method for us to build highly flexible systems, because it offers powerful facilities, enabling technically inexperienced users to address a wide range of problems (e.g. non-functional problems like inappropriate user interfaces, functional problems like missing functionality) by adapting software systems on their own, while the system is already in use. A good introduction to end user development can be found in Sutcliffe & Mehandjiev (2004) or Lieberman et al. (2005b).

Gantt and Nardi (1992) and Mackay (1990) characterize three different types of end users in their studies with different technical skills. In our case end users could be characterized as local developers, who are domain experts and not professional programmers and who adapt the organizations' software to their requirements. To give these people a better support, we are building appropriate adaptation tools for end users in our current research project (see section 3). Moreover we are studying the phenomena of inappropriate software systems in small and medium sized enterprises (*SME*) to make these tools corresponding to users' needs. The first point of our research agenda was the development of an understanding of what exactly end users contribute to the development of IT infrastructures in SME. Therefore we have done a case study to identify problems end users have with their software systems and how they try to solve them. In this study we found out that since SMEs usually do not have an IT-Department and consequently no dedicated technical experts, it is even harder to design flexible systems for these companies than for bigger companies, which usually have dedicated technical experts for these tasks.

The paper is organized in seven sections, starting with this introduction. Section 2 explains why end user development is important for SME. In section 3 we provide a brief description of our research project to clarify the setting in which our case study took place. After this we give in section 4 a detailed description of the research method, which we used for our case study. Section 5 presents the results of the study from which we have drawn the implications discussed in section 6. In section 7 we finally summarize this paper and give an outlook on our ongoing work.

2 END USER DEVELOPMENT IN SME

Small and medium sized enterprises have to be flexible in order to survive in a global market. Today's organizational environments are often highly dynamic. To deal with these environments successfully, organizations have to adapt their products, work processes and work practices permanently. The same applies for technology, because it usually is a critical factor in business processes. Brynjolfsson and Hitt (1998) argue that most productivity effects do not directly result from computer applications, but can be attributed to the specific appropriation of technology in their organizational settings. A successful appropriation of information technology often leads to new work practices, innovative business processes or modified organizational structures. A number of studies and approaches have been developed to capture the appropriation of information systems as a phenomenon (Chudoba, 1999; Dennis et al., 2001; Jasperson et al., 1999) – but our intention goes beyond it.

We understand appropriation activities at a level that allows designing information systems, which actively support such appropriation activities (Pipek & Syrjänen, 2006). Processes of organizational appropriation take considerable time to find solutions, which need to be developed for specific fields of application (Orlikowski, 1996). Within these processes of organizational adaptation, the introduction of end user development can play an important role as an enabler for product innovation as well as a catalyst for organizational change and the adaptation of existing work processes and work practices. The need of SMEs for highly adaptable standard software products, contributing to their needs, could be met by providing appropriate end user development tools within these software systems. In contrast to larger companies, SME have only a very limited amount of human and financial resources. They struggle with many changes and have to adjust their division of labor quite often. SME's software systems have to match this demand for flexibility, making it harder for software companies to provide appropriate products. Furthermore software companies have to address a very heterogeneous and competitive market with many customers, who cannot afford to buy individual software products. Designing standard software for many different stakeholders is difficult, because it is impossible to meet all of their requirements.

3 BACKGROUND OF THE RESEARCH PROJECT

EUDISMES is a research project within the program "Software Engineering 2006". Three research partners (University of Siegen, SAP Research and Buhl Data GmbH) participate in the project. Besides these research partners, five partners from various industrial areas participate also in the project. The two researching software producers, SAP AG and Buhl Data GmbH, develop business software for small enterprises and in case of SAP enterprise resource planning systems (*ERP systems*). The project focuses on the development of innovative end user development strategies and techniques for the business software market. SMEs are in focus of the project, because developing software for this target group is a big challenge (cf. section 2). End user development can open up new perspectives and enable end users, as non-professional developers, to manage their local infrastructure of information systems. Our collaboration with the industrial project partners is valuable to gain information about ongoing appropriation and adaptation activities. As presented in this paper, empirical research was the first step of the project.

4 RESEARCH METHODOLOGY

For our empirical work we used qualitative research methods (Kvale, 1996). All five industry partners were involved in the study. In a first step we identified the internal organizational structures of the five partner companies and tried to get a general overview. We did this in an exploratory way by means of open semi-structured face-to-face interviews. The interview partners of the first interview session were

the CEOs (*Chief Executive Officer*) of the companies or the managers of the IT departments. In the second interview session we extended the group of participants to "ordinary" employees, and focused on specific topics (cf. section 4.1) to obtain a second perspective of the companies from a different angle. By using qualitative research methods we achieved a very detailed and profound insight into the working field of the participants. We chose semi-structured interviews to grasp the subjective views of the interviewees. This approach made it easy to compare the numerous interviews and allowed us to address specific topics (Kvale, 1996). The design of the interview guides followed the interview criteria proposed by Merton & Kendall (1946):

- Non-directivity (no manipulation of the interviewee)
- Elaboration of the interviewee's subjective meaning of the interview topics
- Grasping a wide range of facets concerning the interview topic
- Emotional profoundness of the interview statements and situations

4.1 Setting

All interviews were conducted at the companies' sites. This approach was chosen on the one hand, because the distances (spatial) between our university and the project partners' company sites are relatively long. On the other hand we wanted to interview the end users in familiar surroundings to keep the de-contextualization from their work space as low as possible. The empirical work was done, as mentioned before, in two sessions. In the first session – accomplished in spring 2006 – we wanted to develop a better understanding of the companies and their organizational forms by asking questions about their organization, their use of technology, etc. Altogether we interviewed seven people. Three of them were CEOs, because they know the structure of their organization very well. The remaining four interviews were done with employees from the IT departments (in most cases the managers) of the companies, because they have knowledge about the companies' IT infrastructure. We used two different interview guidelines for the interviews to be responsive to the expected differences between the participating smaller (less than 30 employees) SME and the larger (more than 100 employees) ones. The two smaller companies are handicraft enterprises with approximately 12 and 26 employees. One of the larger companies is a software company with approximately 500 employees, while the two other companies are industrial companies (production of automotive parts/satchels and bags) with approximately 120 and 155 employees. The interviews were carried out by three interviewers (one person from each research partner). The duration of the interviews was between 80 and 120 minutes. The interviews were recorded with a digital voice-recorder, allowing us to transcribe the interviews later on. The transcription of the interviews was done at our department by well trained researchers. The results of the transcript analysis enabled us to formulate a first draft of customization options for the software systems of SMEs. Furthermore, it helped us to identify relevant interview partners for the second interview session.

In summer 2006 the second empirical session took place. In this phase we wanted to identify specific aspects of the interviewees' work practices, like problem solving strategies and users' demand for adaptations. In contrast to the interviews of the first session, we extended the group of participants. We interviewed a variety of users (altogether 18) from all five companies, including department managers and employees from different departments, such as financial accounting, buying, order management or human resources. In the second phase we used only one interview guideline. This time – unlike in the first session – we did not expect big differences between the users' answers, because we questioned them about their work practices. The interviews of the second phase were conducted by us without the participation of our two other research partners. Each of the 17 interviews (in one interview session we interviewed two people) took between 45 and 90 minutes.

4.2 Design of the Evaluation

This section describes the design of evaluation used in the interview sessions, by emphasizing the second interview session, whose results will be discusses later on (see section 5). All interviews were transcribed abbreviated. We compared all interviews of each single company, structured and consolidated them with the questions of the interview guideline. Afterwards we structured the interview statements with regard to the research questions to ex-post generated categories of answers (Glaser & Strauss, 1967; Pandit, 1996). In the first phase the examination focuses on common aspects to create a general idea of the companies. To achieve this we defined seven categories for the analysis, deriving from the users' descriptions of critical incidents, and regarding the use as well as the configuration of the ERP infrastructure, namely enterprise characteristics, software infrastructure, use of individual software, adaptation of standard software, hardware infrastructure, enterprise hierarchy and division of labor. In the second phase the examination is supposed to focus on more specific aspects (e.g. problem solving strategies, adaptation skills). Therefore we defined 15 categories of analysis, which resulted from the interpretation of the empirical data and users' descriptions of critical incidents. The categories are briefly presented below to give an overview of the scope of the analysis. Two of them (problem types and problem solving strategies) will be discussed in detail in section 5.

- **Roles: -** explains informal roles within the companies. In contrast to formal roles (e.g. organizational position of the employee), informal roles are roles, defined by the subjective role definition of the person concerned and assigned to this person by the colleagues (e.g. colleague X is an expert for Microsoft Excel).
- **Problem types:** - contains all kinds of employees' problems, which could be solved with appropriate end user development techniques (e.g. usability problems).
- **PC user types:** - illustrates the knowledge and abilities of users regarding personal computers (e.g. office user with little knowledge about personal computers).
- **Learning resources:** - focuses on learning resources, which are used by users while learning to use a new software system (e.g. training courses).
- **IT infrastructure awareness:** - expresses which parts of the technical infrastructure are hidden to users (e.g. backup concepts).
- **Influence on infrastructure development:** - portrays which persons could influence the infrastructure development of the companies to which extend (e.g. a manager could decide to buy new software or make suggestions to adapt an existing one).
- **Knowledge about functionality**: - describes how many functions of their software systems users know and use (e.g. users know only the functions they need to do their daily work).
- **Problem solving strategies: -** illustrates users' behavior in case of a problem with their hardware or software (e.g. users ask their colleagues for help).
- **Adaptation: -** describes whether users know that it is possible to adapt software systems and how much they know about the adaptation features of their software systems (e.g. user knows that he could customize his SAP system).
- **Adaptation ratings:** - depicts whether users think that adapting their software systems is beneficial or not (e.g. adaptation is beneficial, because it decreases the time to do a certain task).
- **Adaptation problems:** - explains whether users have problems or think that there are problems to adapt software systems (e.g. adaptations could lead to problems for other users).
- **Adaptation quality:** - what do users think about the quality of their adaptations? (e.g. user's adaptations were all beneficial and successful and have therefore a high quality).
- **Cooperation: -** illustrates whether users cooperate to do an adaptation of a software system (e.g. users of a department cooperate to customize the SAP system).
- **Adaptation Culture: -** does the company encourage its employees to adapt its IT infrastructure? (e.g. company has an open culture, allowing users to adapt their systems).

5 RESULTS

In this section we present the results of two important categories (problem types and problem solving strategies) of our case study, which are relevant for IS design, especially if the system should be adaptable. The analysis of problem types allows IS designers to choose appropriate end user development techniques that should be considered for the implementation of the system, giving users the possibility to reduce the problems they could have with the system. Analyzing the problem solving strategies of users helps IS designers to determine what kind of people are involved in these processes, and to implement appropriate support mechanisms into their systems. There are different end user development techniques to support such community-oriented processes (e.g. delegation support systems). The results of our study are illustrated with some examples[1] of our interviews.

5.1 Problem Types

The term "problem types" refers to all kinds of problems of end users, which could be solved with appropriate end user development concepts. The problems we identified in our interview analysis could be divided into four partially overlapping categories:

- Usability Problems
- Non-functional Problems
- Functional Problems
- Limited Problem Perception

At the end of each subsection we make *suggestions* for each category, how the presented problems could be addressed with appropriate end user development techniques. The techniques are classified by the three levels of end user tailoring: customization, integration and extension, which were introduced by (Mørch, 1997) and also apply to end user development. The methods of these levels demand different technical knowledge and differ in their complexity.

5.1.1 Usability Problems

The first category, usability problems, describes user problems with graphical user interfaces of their software, including unanticipated dialog sequences. Typical are navigation problems within the software or the accessibility of several functions. Some interviewees uttered usage problems with their SAP systems, because they were only able to find a function they needed in a short time if they already knew where they had to search. Otherwise they had to browse through an almost "infinite" tree of functions to find the function they searched for. One user made this statement:

"The SAP menu contains a data tree from which different sub trees can be opened. If I didn't know which transaction I need, and I had to search for it, I would be lost. […] The tree can be opened further and further and is infinite long."

More problems of this category were the graphical user interfaces that are not compliant to users' expectations. One SAP user remarked that the user interfaces of the system are not structured equally. As an example she described her problem with the "confirm symbol":

"What I don't like about SAP is the diverse design of user interfaces. A clear line is not existent. […] You learn that you have to click the check-symbol which means enter. But there is also a clock with check-symbol, which didn't let one get ahead. […] Sometimes the check-symbol is located top left and sometimes it is more centered. […] I think the work was done by different developers."

[1] The examples are freely translated from German to English by the authors.

Problems of this category could be addressed by end user development techniques of the customization level. Therefore the designer has to anticipate possible behavior patterns of end users to give them a variety of possibilities (e.g. interface templates, customizable layouts and menus) to deal with occurring problems on their own.

5.1.2 Non-Functional Problems

Non-functional problems are difficulties which do not refer directly to the functionality of a software system. Our empirical results exposed that comprehension problems of the integrated help systems are the most important problem in this category. Many users reported troubles with the integrated help systems of their software. They don't think much of help systems, because they are written from experts for experts and therefore are incomprehensible for them. This is an example of what we heard:

In reply to a question about the usage of the SAP help system: "I don't want to say anything about that. […] there is probably nothing in it that is comprehensible. The system is probably only written from experts for experts."

The previously described problem could be solved with end user development techniques of the integration level. With these techniques end users could be enabled to adapt and extend systems in accordance to their needs. In case of the help systems users could make it possible to complement the existing systems with self-written texts and change the existing ones.

5.1.3 Functional Problems

In comparison with the just discussed non-functional problems, functional problems can be reduced to the functionality of the software. Our analysis has shown that the main issues of this category are problems with the interface of a program, as well as a limited functional range. Many SAP users described problems with the analysis of data within the SAP system. For example one user struggled to collect the necessary data to do his credit limit checks:

"There are often problems, when I want to compare things. I sometimes have the problem that I have to access four or five thing in order, to get the things I need […] for example the annually credit limit check of our customers. Therefore I need master data and data from the SD, some data from financial accounting and I don't get them by pressing a button."

To overcome the shortcomings of limited functionality, users started to export the relevant data to Microsoft Excel and created their needed reports within this tool. Unfortunately, it is not possible in any case to do an appropriate, automated export from the SAP system to Excel. In this case users had to collect their data within the SAP system manually and copy and paste this data into their Excel sheet. One user described this issue that way:

"Statistics […] my colleague wanted to know, how much overrun each employee of his department had daily, during the last month […] now it would be wonderful, if we were able to transfer this into an Excel file; no Mrs. X has to type each number herself."

Furthermore, some SAP Key Users liked to have more possibilities to tailor their systems without the need of external consultants.

Besides these issues with the SAP systems, we discovered also software products, which do not meet the promised functionality in a particular use case. One user told us about a program, which his company uses to create quotes. This system, called *HWS*, is an optimized standard software product for his particular industry. The program didn't offer him the needed flexibility to do his calculations for a particular quote, because the program didn't allow him to make "a verifiable determination of the needed masses". Therefore he used Excel to do this task, while the HWS system was only used by his colleague to manage and print quotes. The use of Excel for the calculation led to a new problem. It was impossible to exchange data between the HWS system and Excel. As a solution for this the two

colleagues used printouts to exchange data between the HWS system and Excel. The user described the whole process like this:

"When I get the prices back, I open my Excel template and start typing the whole text of the quote – if I have enough time. […] the problem is that this text is not automatically available in the HWS system. Therefore it has to be typed again completely. […] We have to type it at least twice. It would be nice, if we had some kind of program for the HWS system, which is directly connected with my Excel template."

Appropriate end user development techniques for the described problems in this category are techniques of the integration and/or extension level. Users could be enabled to extend parts of the application (e.g. improved analysis functions), connect different systems in an appropriate manner or add completely new functions with these techniques (e.g. appropriate export functions).

5.1.4 Limited Problem Perception

This category refers, in contrast to the prior ones, not directly to specific problems of the users, but to a group of people. We assigned all users, who had – in their eyes – no problems with their software, to this group. These users "arranged" themselves with the existing systems and want to avoid every modification or exchange. They think their systems are monolithic and therefore untouchable. This dialogue illustrates it quite well:

Question (concerning the software systems): "Isn't there sometimes the situation that you get angry about something?" Answer:" Normally not […]." Question: "Let's say someone would install new software on your computer that has a completely different usage […]?" Answer: "I would throw my hands up in horror."

Particular end user development techniques for this problem type do not exist. To overcome these problems, the problem awareness of the users have to be stimulated in a first step to increase their willingness to deal with problems and to lead them to adapt their software systems in a second step.

5.2 Problem Solving Strategies

The second selected topic of our evaluation presents the behavior of users in case of a problem. To structure the various problem solving strategies, we defined them as structured processes (*escalation patterns*). The problem solving behavior of users is rather problem driven, meaning that users select their help sources problem oriented and not sympathy driven (e.g. always ask their best friend for help). The processes which we identified are separated by their problem types. This leads to two different process categories, containing hardware problems on the one hand and software problems on the other.

In this paper we focus on software problems, because the strategies of this category are more interesting for the design of information systems. We visualized the processes as event driven process chains (*EPC*), which were introduced by Keller et al. (1992). Event driven process chains are a structured, widely understood form of presentation for processes and therefore a good choice for the presentation of the identified process solving strategies. Figure 1 shows the EPC that visualizes our empirically discovered problem solving strategies for software problems. The process chain offers many different ways to get from the event at the top (software problem) to one of the events at the bottom. Not all strategies lead to a solution of the problem. To preserve the diverse non-successful tries, we differentiate between three end events of the processes chain.

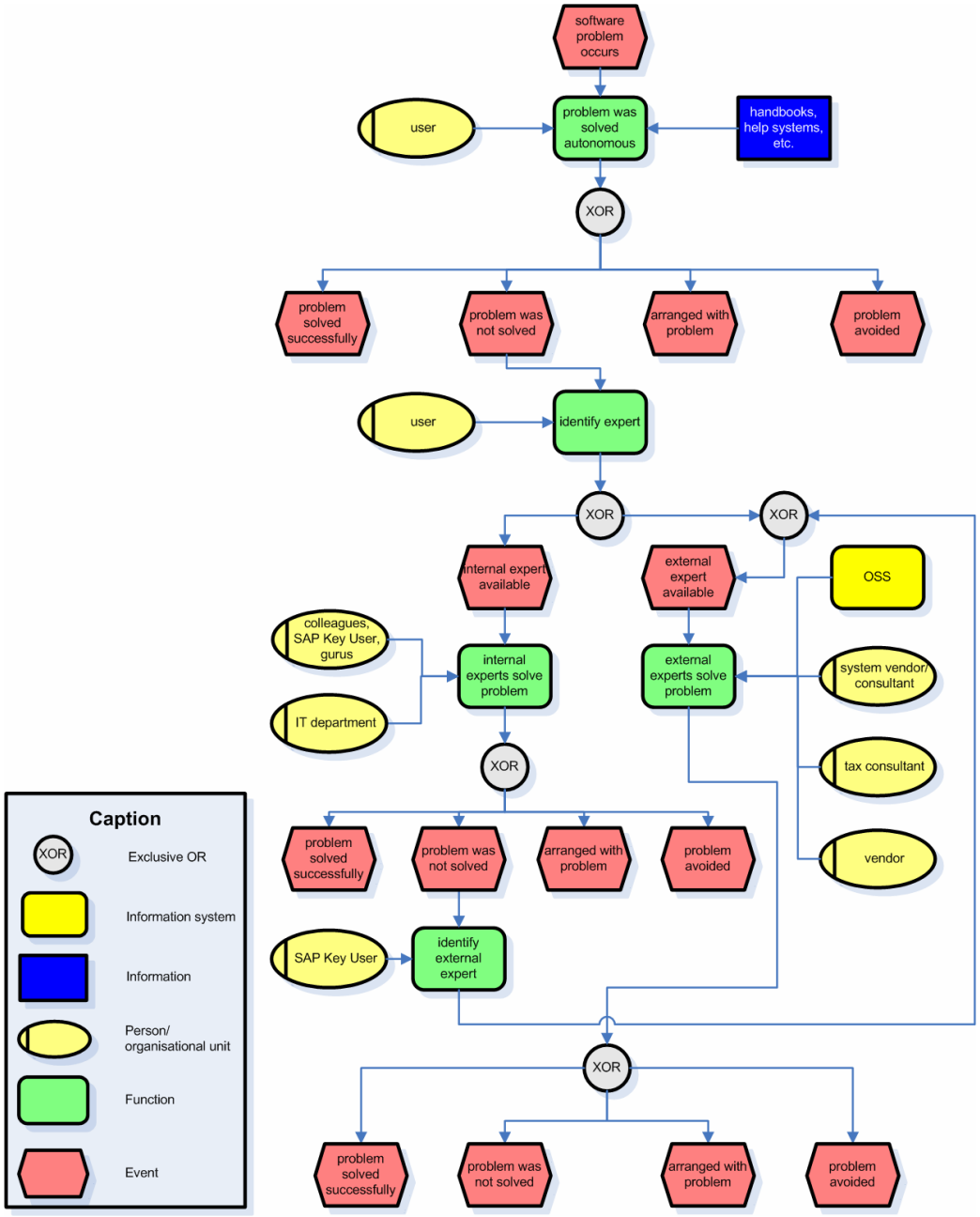

Figure 1 Problem solving strategies in case of a software problem

- *Problem was not solved* (no solution at all): The user was not able to solve the problem. There is no possibility to avoid the problem or to "arrange" with it (e.g. the user needs an export function to Excel; the software vendor offers this functionality in the next software version, which is too expensive to buy for the user).
- *Arranged with problem* (long term "solution"): The user ignores the problem permanently. He/She doesn't want to solve it anymore (e.g. the user has problems to navigate through the software; instead of trying to solve these problems with the offered bookmark function, the problem is ignored).
- *Problem avoided* (short term "solution"): The user applies an alternative way to quickly do his task and therefore avoids the problem (e.g. the user has a problem with his/her software; instead of trying to solve it, he/she uses other materials to do the task).

To give examples of discovered problem solving strategies, we now describe two possible problem solutions of the EPC.

Example 1: At first Karen tries to solve the problem of creating a special KPI (*Key Performance Indicator*) within her SAP system by using its help system. After an hour of searching and browsing through it she surrenders, stops trying and starts to think if anyone else could help her to solve the problem. Preferably it should be a colleague, because otherwise the company has to pay for the external support. Maybe her colleague John could help. He is the SAP Key User of her department and therefore knows a lot about the SAP system. She calls John and tries to solve the problem together with him. After ten minutes John comes to the conclusion that it's useless to continue trying. Neither he, nor another SAP Key User of the company or the IT department could solve the problem easily. But maybe an external expert could help. He requests Karen to wait for a while and reflects who could be the right person for solving this problem or if it would be necessary to write an OSS (*Online Support System*) ticket. He comes to the conclusion that Mr. White, who is one of the companies' SAP consultants, is the right one and starts to write him an email. Mr. White replies after some hours with a proper solution in which he explains how the KPI could be created.

Example 2: Margaret tries to create a large quote for a customer. After entering 64 positions of the quote, the system offers her no new input field for the next position. She tries to work out a proper solution on her own without using handbooks or the help system of the software, because she doesn't like them. After some minutes she decides to call the hotline of the software vendor, because she is the only user of the system in the whole company and therefore none of her colleagues could help her. Calling the hotline leads to no solution. Mr. Black, employee of the software vendor, told her that they will increase the number of positions in the next version of their software.

6 IMPLICATIONS

In this section we briefly discuss the implications of our empirical results for an end user oriented design of information systems. To support the problem solving strategies of section 5.2 properly, we propose to consider the succeeding aspects during the design of information systems. Users should be supported to solve problems and to do adaptations corporately. Therefore they should have the possibility to communicate with each other via the system to exchange adaptation configurations, search for configurations and to delegate adaptation tasks to other persons (e.g. colleagues, experts and SAP Key Users). The delegation support should follow common escalation patterns of the organization to ease the system's use for users. This strategy leads to an increased transparency of the whole improvement process of the software as it can be seen in open source projects. A shared repository, containing adaptation configurations and a delegation mechanism could be used to support this kind of corporative end user development. Such a repository could help users' to solve problems to a certain extend, because it supports the employees of a company to do adaptations and could be used as well to involve external consultants into the process more closely. The introduction of such a system demands an open adaptation culture within the company, where employees work together to solve their problems. The problem types that we identified in section 5.1 should be considered as well, when designing information systems. Considering the problem types could increase the quality of information systems' adaptation techniques. To overcome usability problems, the elements of the user interface must be made configurable by the designers. Besides these problems, users uttered navigation problems. These could be addressed by user definable dialogue sequences, which correspond to their work processes and could be started by clicking on a button, etc. To solve the mentioned problems with the help system of a software system, designers should enable users to change and extend existing help texts. The solution of functional problems is even more complicated to establish than the previously described solutions. To overcome data exchange problems between two or more systems, users must be enabled to "program" adequate interfaces in an easy way. More or less the same applies to missing functions of the software. Users must have end user oriented techniques to adapt and program their systems. Possible concepts could be visual or natural

programming techniques or programming by example. The proposed end user development techniques for the variety of problems differ in their complexity and therefore address end users with different technical skills. Configurable user interfaces are wide spread in today's software systems (e.g. configurable toolbars, customizable menus) and are comparably easy to use. More complex adaptation possibilities, like extendable systems, require a higher technical knowledge, but enable users to adapt their system in accordance to organizational needs, as long as the adaptations are not too fundamental. Allowing users to adapt their systems is problematic, because it increases their daily work load and demands that they have to deal with things, which are not directly connected to their job. Furthermore they have to spend some time for the adaptations without receiving a direct benefit, because adaptations usually pay off in the long run.

7 SUMMARY AND FURTHER STUDY

The presented case study was the first step on the research agenda of our current research project ***EUDISMES*** in which we develop innovative end user development techniques for the business software market. We used interviews as a qualitative research method to get a very detailed and profound insight into the working field of our interview partners, which came from various industries. For our analysis of the empirical results we created several categories to focus on specific aspects. Two important categories (problem types and problem solving strategies) of the case study were presented in detail in this paper, because these categories are particularly relevant for the design of adaptable information systems, and could be achieved by end user development techniques. These empirical results were used for the development of suggestions for IS design, to create systems that are not only flexible, but also allow technically inexperienced users to manage the "last mile" of software development towards the establishment of a successful software usage within their use context. This may involve a better communication between the developer and the user context as well as functionality for appropriation support. The establishment and adaptation of ERP systems are often studied from a managerial perspective focusing on business processes and organizational structures. We want to complement these studies with a perspective that concentrates on activities and lived practices of users.

The next step of our research will be the development and integration of end user development tools based on the implications presented in section 6 to prove our suggestions in practice. A close collaboration with the industrial partners, who also contributed to this study, will ensure a high relevance for SMEs practice.

8 ACKNOWLEDGEMENTS

We would like to thank Markus Rohde, Gunnar Stevens and Volker Wulf for invaluable discussions on the issues of end user development in general and the design of the study in particular. We also thank our colleagues, who were involved in the work with the industrial partners. This research was funded by the Deutsche Forschungsgemeinschaft within the SFB/FK 615 'Medienumbrüche' and the German Federal Ministry of Education and Research within the EUDISMES project.

References

Brynjolfsson, E. and Hitt, L. M. (1998) Beyond the productivity paradox. *Commun. ACM*, ACM Press, pp 49-55.

Chudoba, K. M. (1999) Appropriations and patterns in the use of group support systems. *SIGMIS Database*, ACM Press, pp 131-148.

Dennis, A. R., Wixom, B. H. and Vandenberg, R. J. (2001) Understanding fit and appropriation effects in group support systems via meta-analysis. *MIS Quarterly*, pp 167-194.

Desanctis, G. and Poole, M. S. (1994). Capturing the complexity in advanced technology use: Adaptive structuration theory. *Organization Science*, pp 121-147.
Fischer, G. (2006) Software engineering themes for the future. *ICSE '06: Proceeding of the 28th international conference on Software engineering*, ACM Press, New York, NY, USA, pp 1043-1044.
Fischer, G., Giaccardi, E., Ye, Y., Sutcliffe, A. G. and Mehandjiev, N. (2004) Meta-design: A manifesto for end-user development. *Commun. ACM*, ACM Press, pp 33-37.
Gantt, M. and Nardi, B. A. (1992) Gardeners and gurus: Patterns of cooperation among cad users. In *CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems*, pp 107-117, ACM Press, New York, NY, USA.
Glaser, B. G. and Strauss, A. (1967) *The discovery of grounded theory.* Aldine Transaction, New York.
Jasperson, J., Sambamurthy, V. and Zmud, R. W. (1999) Social influence and individual it use: Unraveling the pathways of appropriation moves. *ICIS '99: Proceeding of the 20th international conference on Information Systems*, Association for Information Systems, Atlanta, GA, USA, pp 113-118.
Keller, G., Nüttgens, M. and Scheer, A. W. (1992) Semantische Prozessmodellierung auf der Grundlage ereignisgesteuerter Prozessketten (EPK). *Veröffentlichungen des Instituts für Wirtschaftsinformatik*, pp 1-31.
Kvale, S. (1996) *Interviews: An introduction to qualitative research interviewing.* Sage Publications, Thousand Oaks.
Lieberman, H., Paternó, F., Klann, M. and Wulf, V. (2005a) End-user development: An emerging paradigm. In *End-user development* (Lieberman, H. and Paterno, F. and Wulf, V., Eds), pp 9-15, Springer Netherlands, Berlin.
Lieberman, H., Paternò, F. and Wulf, V. (2005b) *End user development.* Springer Netherlands, Berlin.
Mackay, W. E. (1990) Patterns of sharing customizable software. In *CSCW '90: Proceedings of the 1990 ACM conference on Computer-supported cooperative work*, pp 209-221, ACM Press, New York, NY, USA.
Maclean, A., Carter, K., Lövstrand, L. and Moran, T. (1990) User-tailorable systems: Pressing the issues with buttons. In *CHI '90: Proceedings of the SIGCHI conference on Human factors in computing systems*, pp 175-182, ACM Press, New York, NY, USA.
Merton, R. K. and Kendall, P. L. (1946) The focused interview. *American Journal of Sociology* 51, 541-557.
Mørch, A. (1997) Three levels of end-user tailoring: Customization, integration and extension. (Kyng, M. and Mathiassen, L., Eds), pp 51-76, The MIT Press, Cambridge, MA.
Nardi, B. A. (1993) *A small matter of programming: Perspectives on end user computing.* MIT Press, Cambridge, MA.
Orlikowski, W. J. (1996) Evolving the notes: Organizational change around groupware technology. In *Groupware and teamwork* (C.Ciborra, Ed), pp 23-59, J. Wiley, Chichester.
Pandit, N. R. (1996) The creation of theory: A recent application of the grounded theory model. *The Qualitative Report* 2,
Pipek, V. and Syrjänen, A. L. (2006) Infrastructuring as capturing in-situ design.
Stevens, G., Quaisser, G. and Klann, M. (2005) Breakin it up: An industrial case study of component-based tailorable software design. In *End user development* (Lieberman, H. and Paternò, F. and Wulf, V., Eds), pp 269-294, Springer Netherlands, Berlin.
Stiemerling, O., Kahler, H. and Wulf, V. (1997) How to make software softer — designing tailorable applications. *DIS '97: Proceedings of the conference on Designing interactive systems*, ACM Press, New York, NY, USA, pp 365-376.
Suchman, L. (1987) *Plans and situated action.* Cambridge University Press, Cambridge, UK.
Sutcliffe, A. and Mehandjiev, N. (2004) End user development. *Special Issue of the Communications of the ACM*, pp 31-32.
Von Hippel, E. and Katz, R. (2002) Shifting innovation to users via toolkits. *Manage. Sci.* 48 (7), 821-833.